Multi-Model Analysis - Strategies for Stabilizing Tomato Prices in Nepal

Felicia D. O'Garro

2024-12-11

Strategies for Stabilizing Tomato Prices in Nepal

```
#Load necessary libraries

library(strucchange)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: sandwich
```

```
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(stringr)
```

```
##
## Attaching package: 'stringr'
```

```
## The following object is masked from 'package:strucchange':
##
##     boundary
```

```
library(tidyr)
library(ggplot2)
library(mice)
```

```
##
```

```
## Attaching package: 'mice'

## The following object is masked from 'package:stats':
##
##     filter

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(quantmod)
```

```
## Loading required package: xts

##
## ######################### Warning from 'xts' package ##########################
## #                                                                             #
## # The dplyr lag() function breaks how base R's lag() function is supposed to  #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or       #
## # source() into this session won't work correctly.                            #
## #                                                                             #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop           #
## # dplyr from breaking base R's lag() function.                                #
## #                                                                             #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning.  #
## #                                                                             #
## ###############################################################################

##
## Attaching package: 'xts'

## The following objects are masked from 'package:dplyr':
##
##     first, last

## Loading required package: TTR
```

```
library(forecast)
library(vars)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select

## Loading required package: urca

## Loading required package: lmtest
```

```r
library(lmtest)
library(corrplot)
```

```
## corrplot 0.94 loaded
```

```r
library(MARSS)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
library(fastDummies)
library(forecast)
library(rugarch)
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'rugarch'
```

```
## The following object is masked from 'package:mice':
##
##     convergence
```

```
## The following object is masked from 'package:stats':
##
##     sigma
```

```r
library(urca)
library(ggbiplot)
library(cluster)
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-8
```

```
library(zoo)
```

Exploratory Analysis

Loading and cleaning data and preliminary data analysis

```
#Load data

data <- read.csv("kalimati_tarkari_dataset.csv")
data$Date <- as.Date(data$Date)
data$Average <- as.numeric(data$Average)
data$Minimum <- as.numeric(data$Minimum)
data$Maximum <- as.numeric(data$Maximum)

# Check for missing values
sum(is.na(data))
```

```
## [1] 0
```

```
# Remove rows with missing values
data <- na.omit(data)

head(data)
```

```
##   SN           Commodity       Date Unit Minimum Maximum Average
## 1  0  Tomato Big(Nepali) 2013-06-16   Kg      35      40    37.5
## 2  1 Tomato Small(Local) 2013-06-16   Kg      26      32    29.0
## 3  2          Potato Red 2013-06-16   Kg      20      21    20.5
## 4  3        Potato White 2013-06-16   Kg      15      16    15.5
## 5  4  Onion Dry (Indian) 2013-06-16   Kg      28      30    29.0
## 6  5       Carrot(Local) 2013-06-16   Kg      30      35    32.5
```

```
summary_stats <- data.frame(
  Metric = c("Minimum", "Maximum", "Average"),
  Mean = c(mean(data$Minimum, na.rm = TRUE), mean(data$Maximum, na.rm = TRUE), mean(data$Average, na.rm
  SD = c(sd(data$Minimum, na.rm = TRUE), sd(data$Maximum, na.rm = TRUE), sd(data$Average, na.rm = TRUE)
)

print(summary_stats)
```

```
##    Metric     Mean       SD
## 1 Minimum 85.42394 77.05890
## 2 Maximum 94.16128 82.37586
## 3 Average 89.79261 79.61900
```

```
# Compute correlation for each group (e.g., by Commodity)
grouped_corr <- data %>%
  group_by(Commodity) %>%
  summarise(correlation = cor(Minimum, Maximum, use = "pairwise.complete.obs"))
```

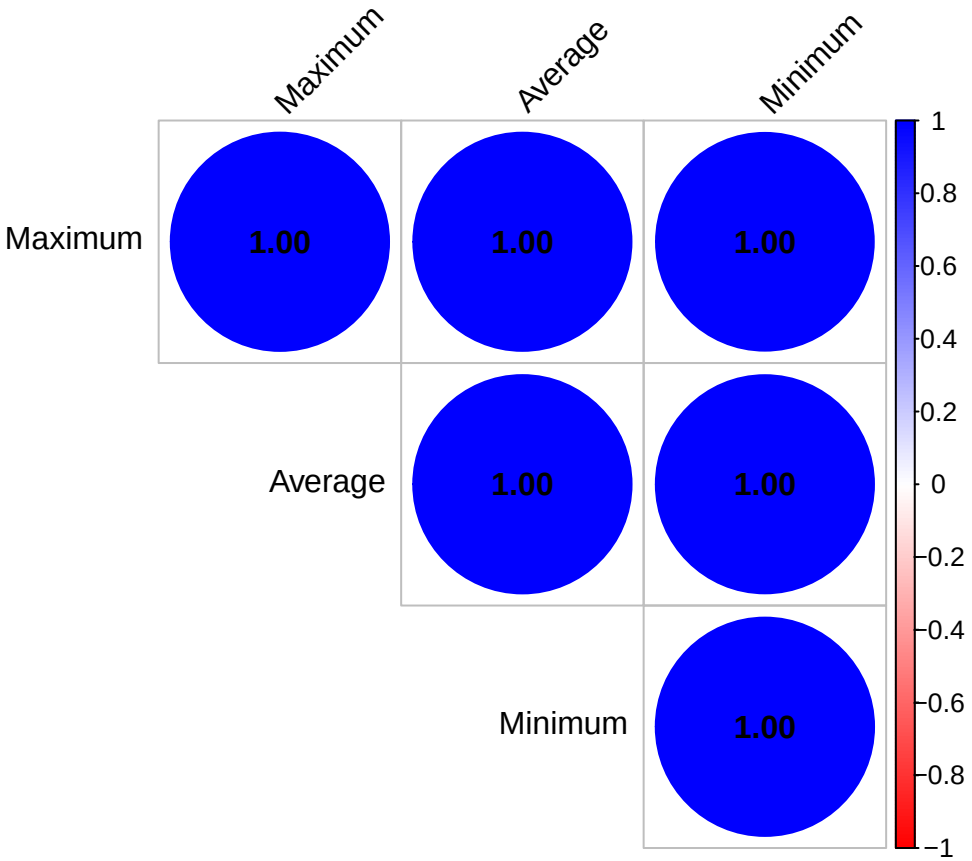
```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `correlation = cor(Minimum, Maximum, use =
```

```
##   "pairwise.complete.obs")`.
## i In group 72: `Commodity = "Mango(Chousa)"`.
## Caused by warning in `cor()`:
## ! the standard deviation is zero
```

```
print(grouped_corr)
```

```
## # A tibble: 132 x 2
##    Commodity       correlation
##    <chr>                 <dbl>
##  1 Apple(Fuji)           0.962
##  2 Apple(Jholey)         0.994
##  3 Arum                  0.972
##  4 Asparagus             0.990
##  5 Bakula                0.994
##  6 Bamboo Shoot          0.946
##  7 Banana                0.997
##  8 Barela                0.988
##  9 Bauhania flower       0.995
## 10 Bitter Gourd          0.997
## # i 122 more rows
```

```
cor_data <- cor(data[, c("Minimum", "Maximum", "Average")], use = "pairwise.complete.obs")

corrplot(cor_data,
         method = "circle",        # Use circles for correlation values
         type = "upper",           # Show only the upper triangle
         order = "AOE",            # Alphabetical order for labels
         tl.col = "black",         # Black text labels
         tl.srt = 45,              # Rotate text labels
         addCoef.col = "black",    # Add coefficients on the plot
         col = colorRampPalette(c("red", "white", "blue"))(200))  # Custom color scale
```



Explore unique commodities

```
# Explore unique commodities

data$Commodity <- as.factor(data$Commodity)
unique_commodities <- levels(data$Commodity)
print(unique_commodities)
```

```
##   [1] "Apple(Fuji)"           "Apple(Jholey)"         "Arum"
##   [4] "Asparagus"             "Bakula"                "Bamboo Shoot"
##   [7] "Banana"                "Barela"                "Bauhania flower"
##  [10] "Bitter Gourd"          "Bottle Gourd"          "Brd Leaf Mustard"
##  [13] "Brinjal Long"          "Brinjal Round"         "Brocauli"
##  [16] "Cabbage"               "Cabbage(Local)"        "Cabbage(Terai)"
##  [19] "Capsicum"              "Carrot(Local)"         "Carrot(Terai)"
##  [22] "Cauli Local"           "Cauli Local(Jyapu)"    "Cauli Terai"
##  [25] "Celery"                "Chilli Dry"            "Chilli Green"
##  [28] "Chilli Green(Akbare)"  "Chilli Green(Bullet)"  "Chilli Green(Machhe)"
##  [31] "Christophine"          "Clive Dry"             "Clive Green"
##  [34] "Coriander Green"       "Cow pea(Long)"         "Cowpea(Short)"
##  [37] "Cress Leaf"            "Cucumber(Hybrid)"      "Cucumber(Local)"
##  [40] "Drumstick"             "Fennel Leaf"           "Fenugreek Leaf"
##  [43] "Fish Fresh"            "Fish Fresh(Bachuwa)"   "Fish Fresh(Chhadi)"
##  [46] "Fish Fresh(Mungari)"   "Fish Fresh(Rahu)"      "French Bean(Hybrid)"
##  [49] "French Bean(Local)"    "French Bean(Rajma)"    "Garlic Dry Chinese"
##  [52] "Garlic Dry Nepali"     "Garlic Green"          "Ginger"
##  [55] "Grapes(Black)"         "Grapes(Green)"         "Green Peas"
##  [58] "Guava"                 "Gundruk"               "Jack Fruit"
```

```
##  [61] "Kinnow"                "Kiwi"                   "Knolkhol"
##  [64] "Lemon"                 "Lettuce"                "Lime"
##  [67] "Litchi(Indian)"        "Litchi(Local)"          "Maize"
##  [70] "Mandarin"              "Mango(Calcutte)"        "Mango(Chousa)"
##  [73] "Mango(Dushari)"        "Mango(Maldah)"          "Mint"
##  [76] "Mombin"                "Mushroom(Button)"       "Mushroom(Kanya)"
##  [79] "Musk Melon"            "Mustard Leaf"           "Neuro"
##  [82] "Okara"                 "Onion Dry (Chinese)"    "Onion Dry (Indian)"
##  [85] "Onion Green"           "Orange(Indian)"         "Orange(Nepali)"
##  [88] "Papaya(Indian)"        "Papaya(Nepali)"         "Parseley"
##  [91] "Pear(Chinese)"         "Pear(Local)"            "Pineapple"
##  [94] "Pointed Gourd(Local)"  "Pointed Gourd(Terai)"   "Pomegranate"
##  [97] "Potato Red"            "Potato Red(Indian)"     "Potato Red(Mude)"
## [100] "Potato White"          "Pumpkin"                "Raddish Red"
## [103] "Raddish White(Hybrid)" "Raddish White(Local)"   "Red Cabbbage"
## [106] "Smooth Gourd"          "Snake Gourd"            "Soyabean Green"
## [109] "Spinach Leaf"          "Sponge Gourd"           "Squash(Long)"
## [112] "Squash(Round)"         "Strawberry"             "Sugarbeet"
## [115] "Sugarcane"             "Sweet Lime"             "Sweet Orange"
## [118] "Sweet Potato"          "Sword Bean"             "Tamarind"
## [121] "Tofu"                  "Tomato Big(Indian)"     "Tomato Big(Nepali)"
## [124] "Tomato Small(Indian)"  "Tomato Small(Local)"    "Tomato Small(Terai)"
## [127] "Tomato Small(Tunnel)"  "Turnip"                 "Turnip A"
## [130] "Water Melon(Dotted)"   "Water Melon(Green)"     "Yam"
```

Create category and variety categories

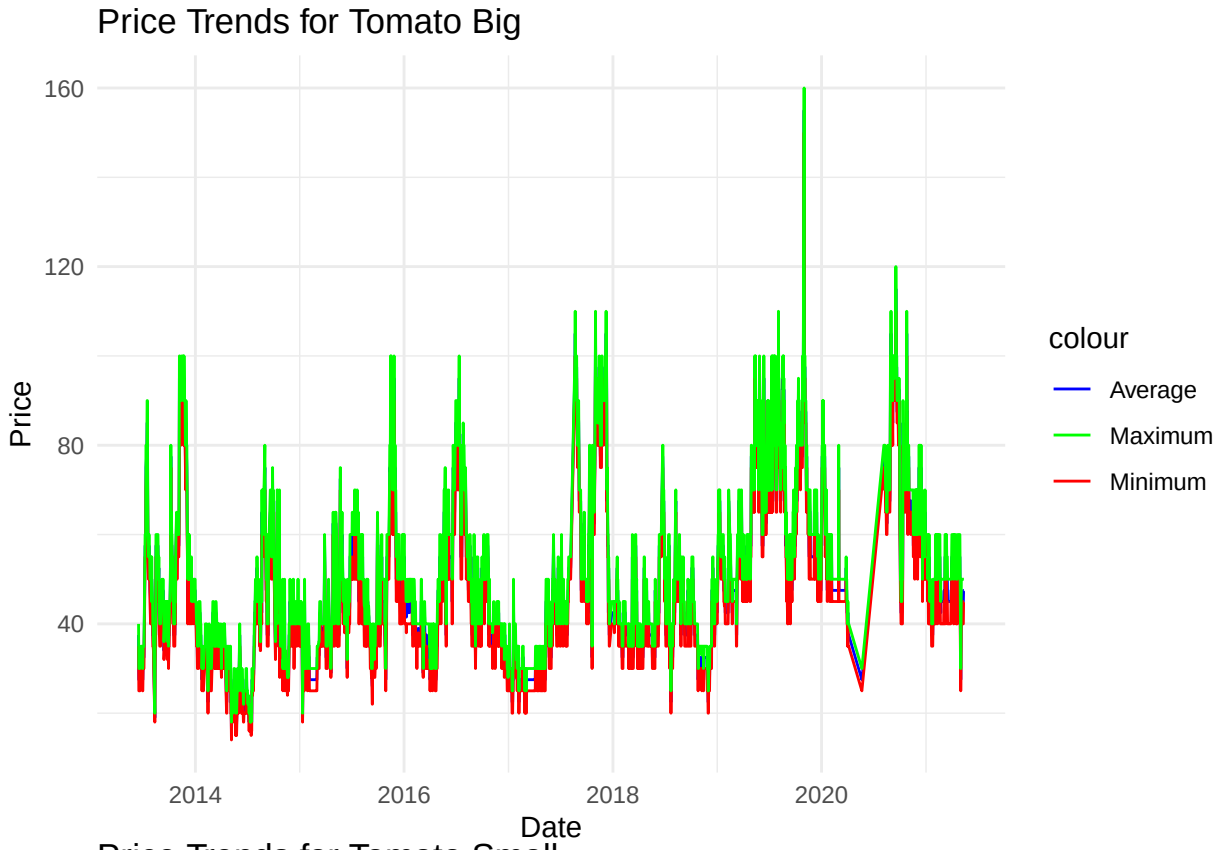
```
# create category and variety categories

data <- data %>%
  mutate(
    Category = case_when(
      grepl("Tomato Big", Commodity) ~ "Tomato Big",
      grepl("Tomato Small", Commodity) ~ "Tomato Small",
      grepl("Potato Red", Commodity) ~ "Potato Red",
      grepl("Potato White", Commodity) ~ "Potato White",
      grepl("Onion Dry", Commodity) ~ "Onion Dry",
      grepl("Onion Green", Commodity) ~ "Onion Green",
      grepl("Apple", Commodity) ~ "Apple",
      grepl("Orange", Commodity) ~ "Orange",
      grepl("Papaya", Commodity) ~ "Papaya",
      grepl("Pear", Commodity) ~ "Pear",
      TRUE ~ "Other"
    ),
    Variety = case_when(
      grepl("Nepali", Commodity) ~ "Nepali",
      grepl("Local", Commodity) ~ "Local",
      grepl("Indian", Commodity) ~ "Indian",
      grepl("Terai", Commodity) ~ "Terai",
      grepl("Tunnel", Commodity) ~ "Tunnel",
      grepl("Mude", Commodity) ~ "Mude",
      grepl("Chinese", Commodity) ~ "Chinese",
      grepl("Fuji", Commodity) ~ "Fuji",
      grepl("Jholey", Commodity) ~ "Jholey",
```

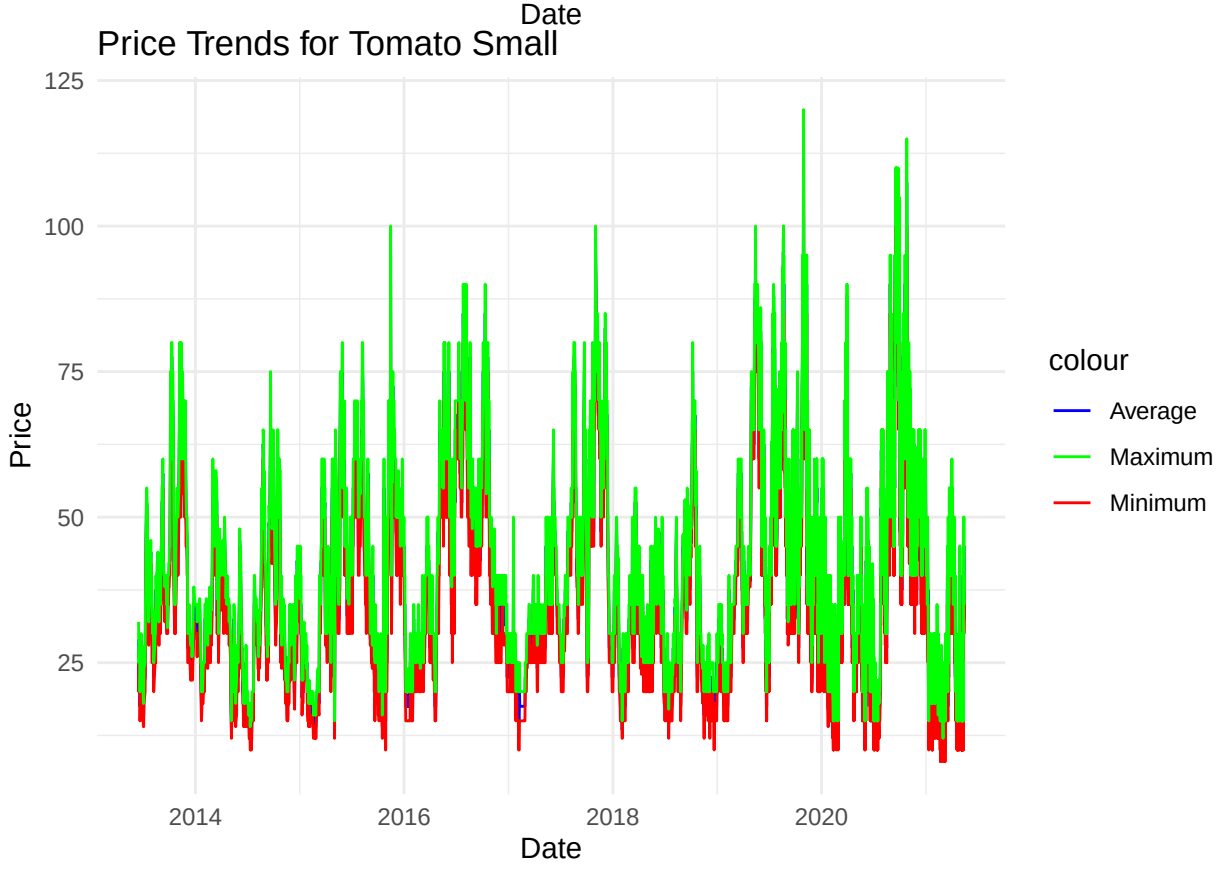
```
      TRUE ~ "Other"
    )
  )
head(data)
```

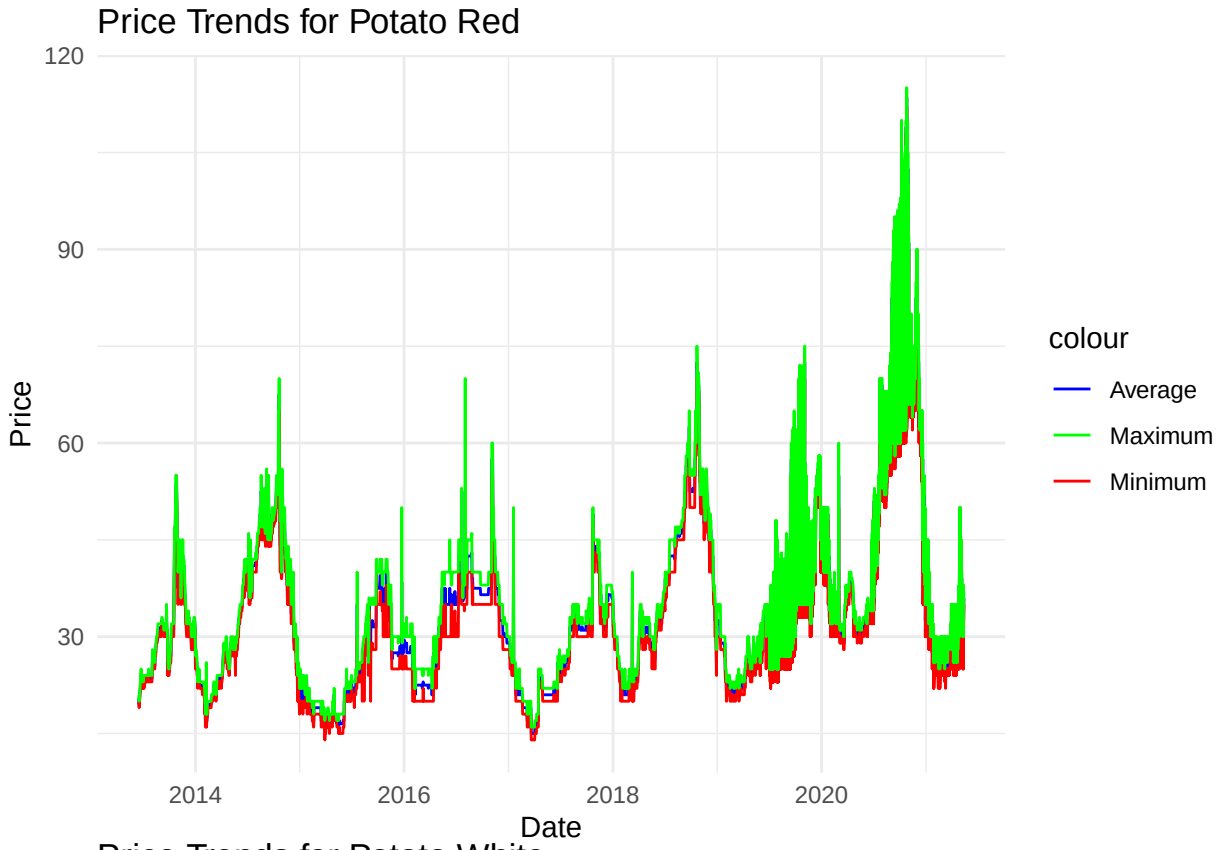
```
##   SN           Commodity       Date Unit Minimum Maximum Average     Category
## 1  0  Tomato Big(Nepali) 2013-06-16   Kg      35      40    37.5   Tomato Big
## 2  1 Tomato Small(Local) 2013-06-16   Kg      26      32    29.0 Tomato Small
## 3  2          Potato Red 2013-06-16   Kg      20      21    20.5   Potato Red
## 4  3        Potato White 2013-06-16   Kg      15      16    15.5 Potato White
## 5  4  Onion Dry (Indian) 2013-06-16   Kg      28      30    29.0    Onion Dry
## 6  5       Carrot(Local) 2013-06-16   Kg      30      35    32.5        Other
##   Variety
## 1  Nepali
## 2   Local
## 3   Other
## 4   Other
## 5  Indian
## 6   Local
```

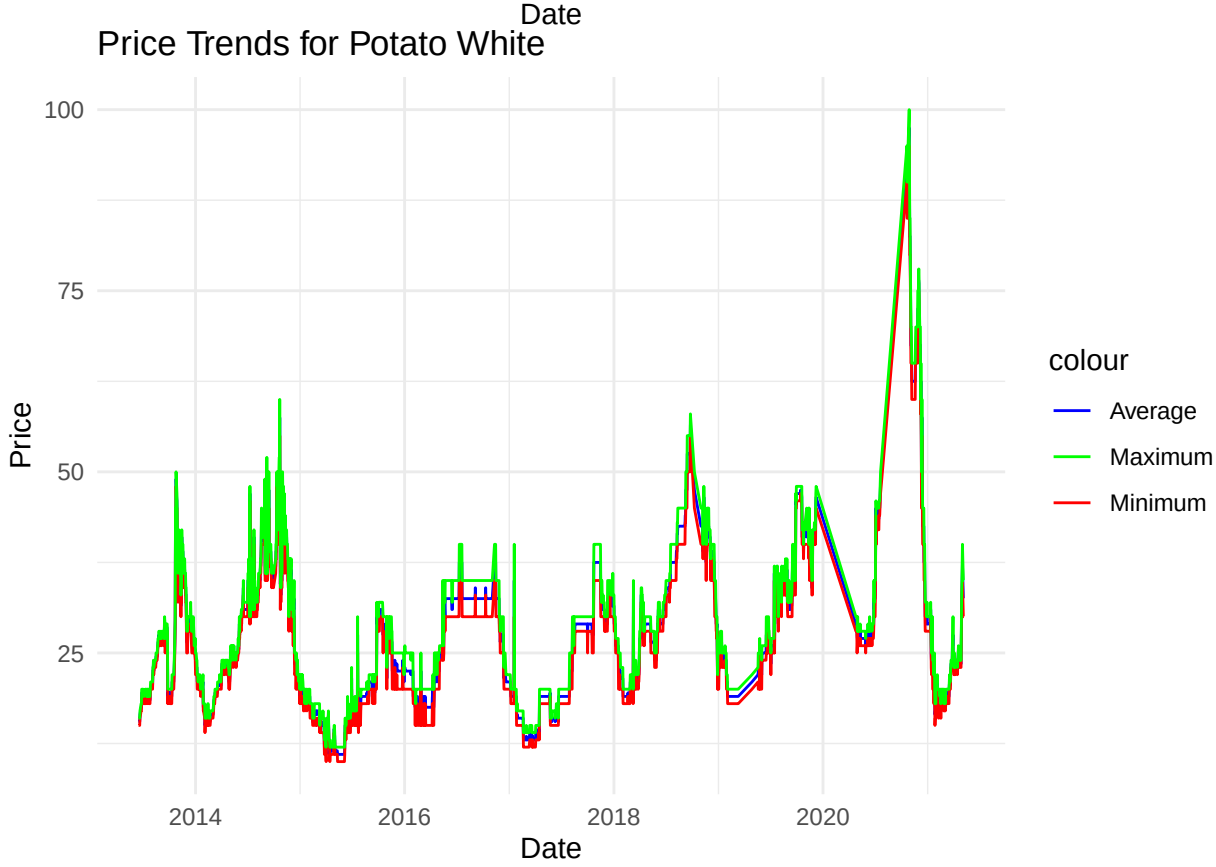
Plots for Categories

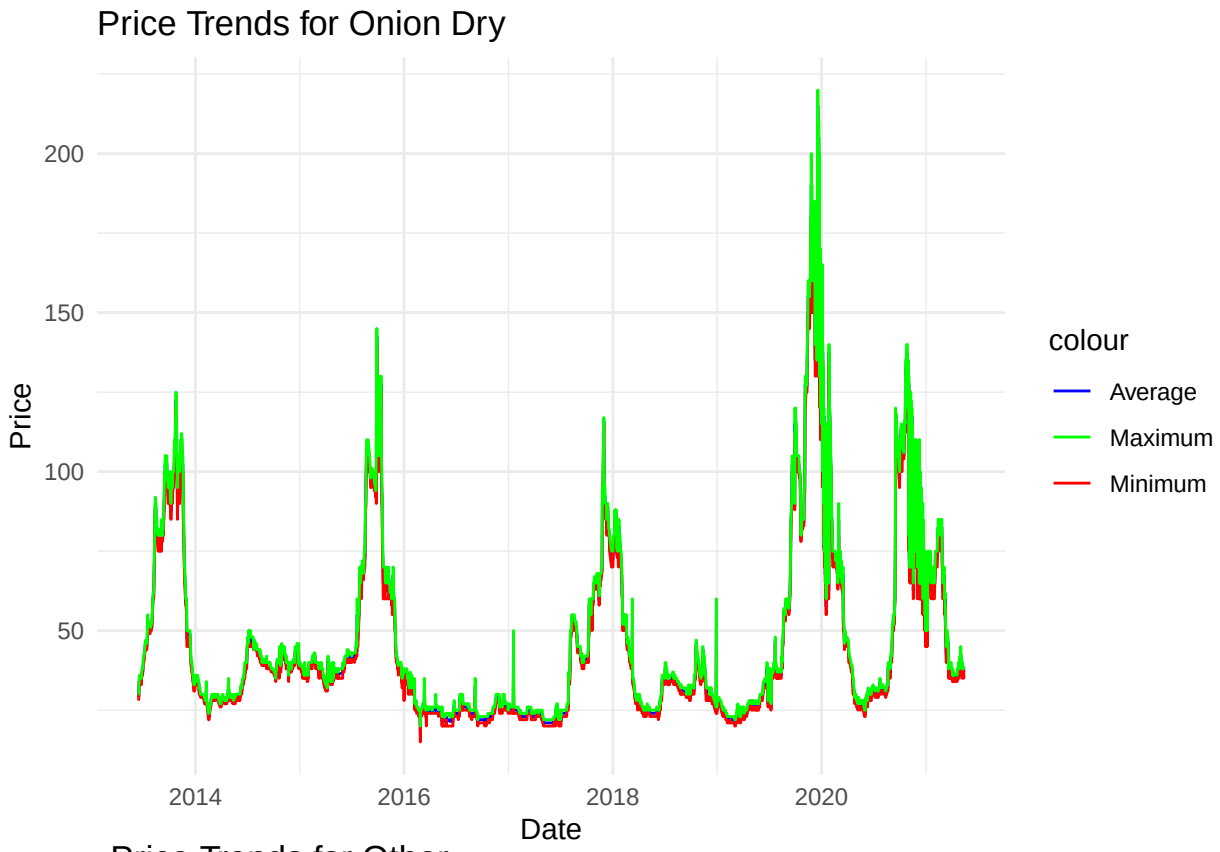
```
# Plot charts for different categories

create_category_plot <- function(category_name) {
  data_filtered <- data %>% filter(Category == category_name)

  ggplot(data_filtered, aes(x = Date)) +
    geom_line(aes(y = Average, color = "Average")) +
    geom_line(aes(y = Minimum, color = "Minimum")) +
    geom_line(aes(y = Maximum, color = "Maximum")) +
    labs(title = paste("Price Trends for", category_name),
         x = "Date",
         y = "Price") +
    scale_color_manual(values = c("Average" = "blue", "Minimum" = "red", "Maximum" = "green")) +
    theme_minimal()
}

unique_categories <- unique(data$Category)

for (category in unique_categories) {
  plot <- create_category_plot(category)
  print(plot)
}
```

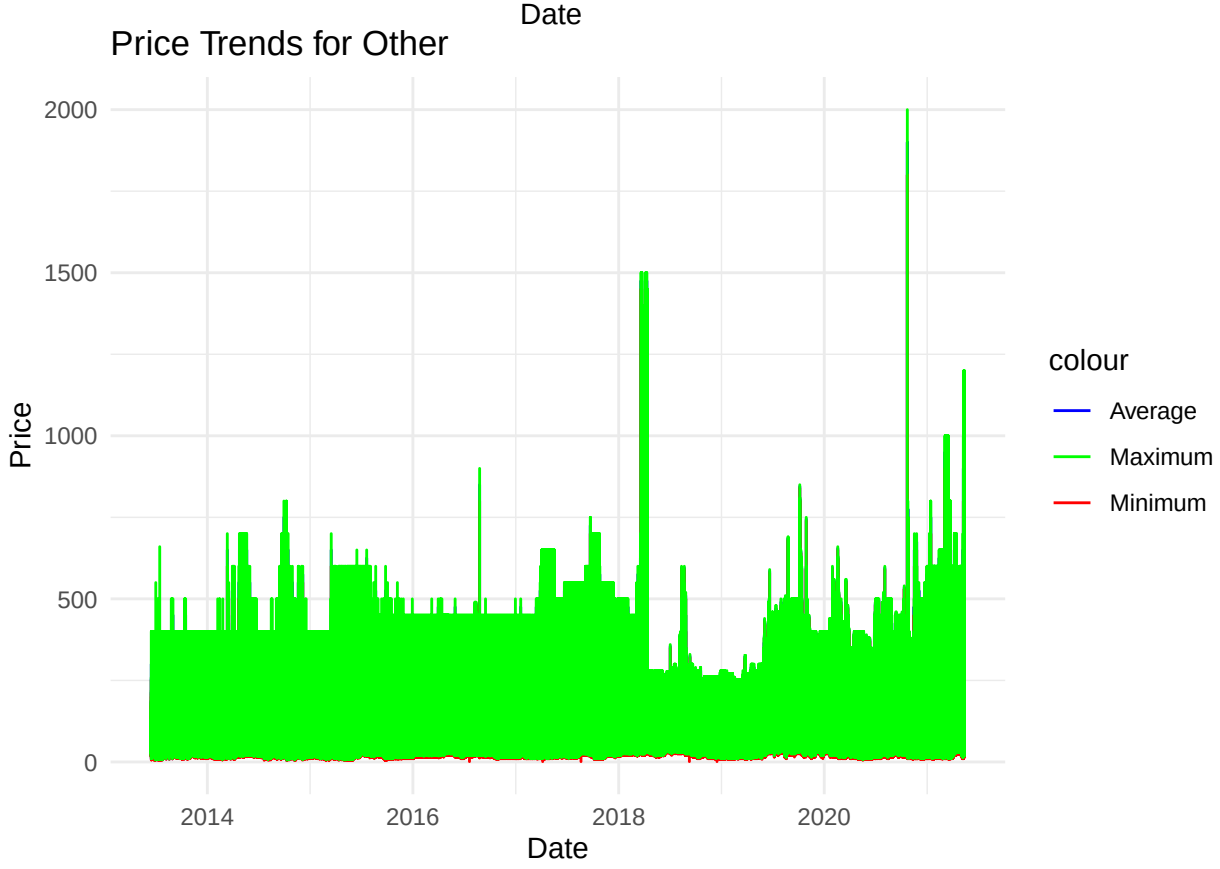

Price Trends for Tomato Big
160
120
80
40
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

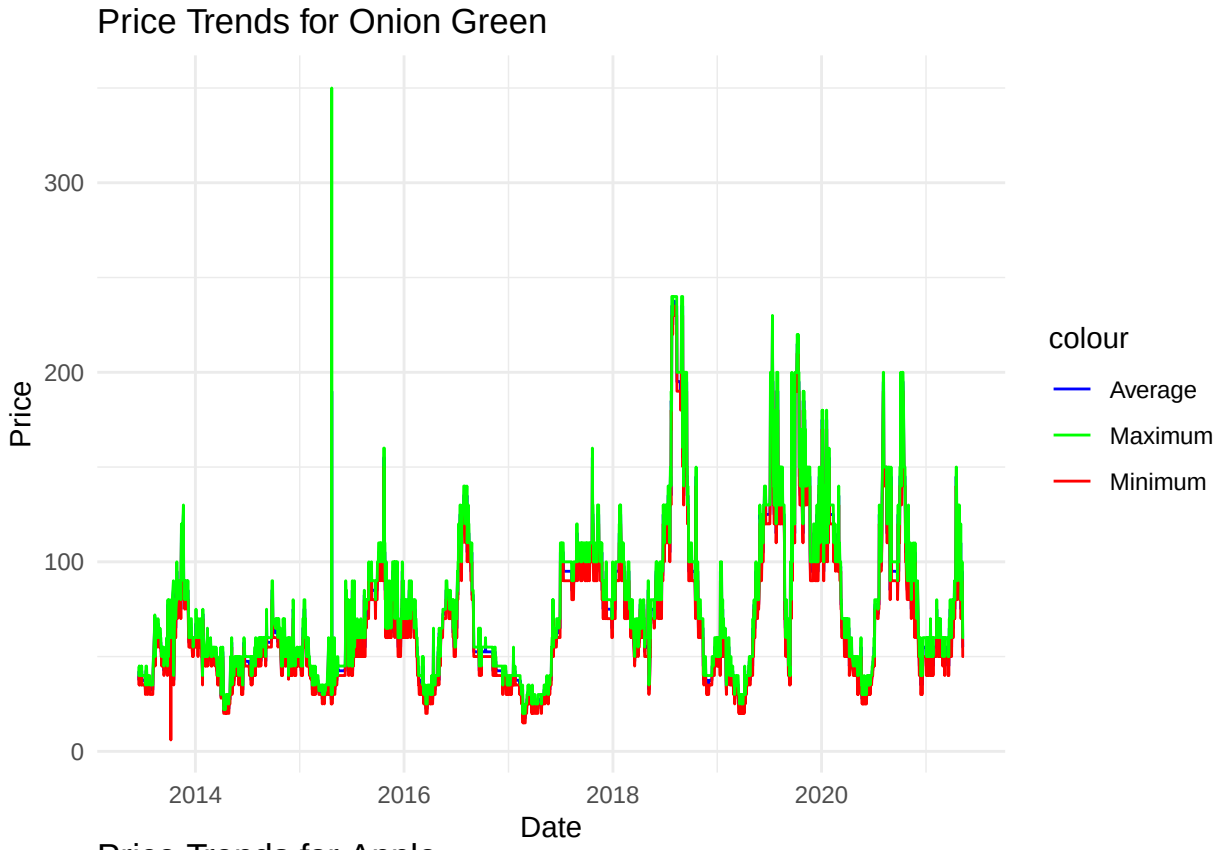

Price Trends for Tomato Small
125
100
75
50
25
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

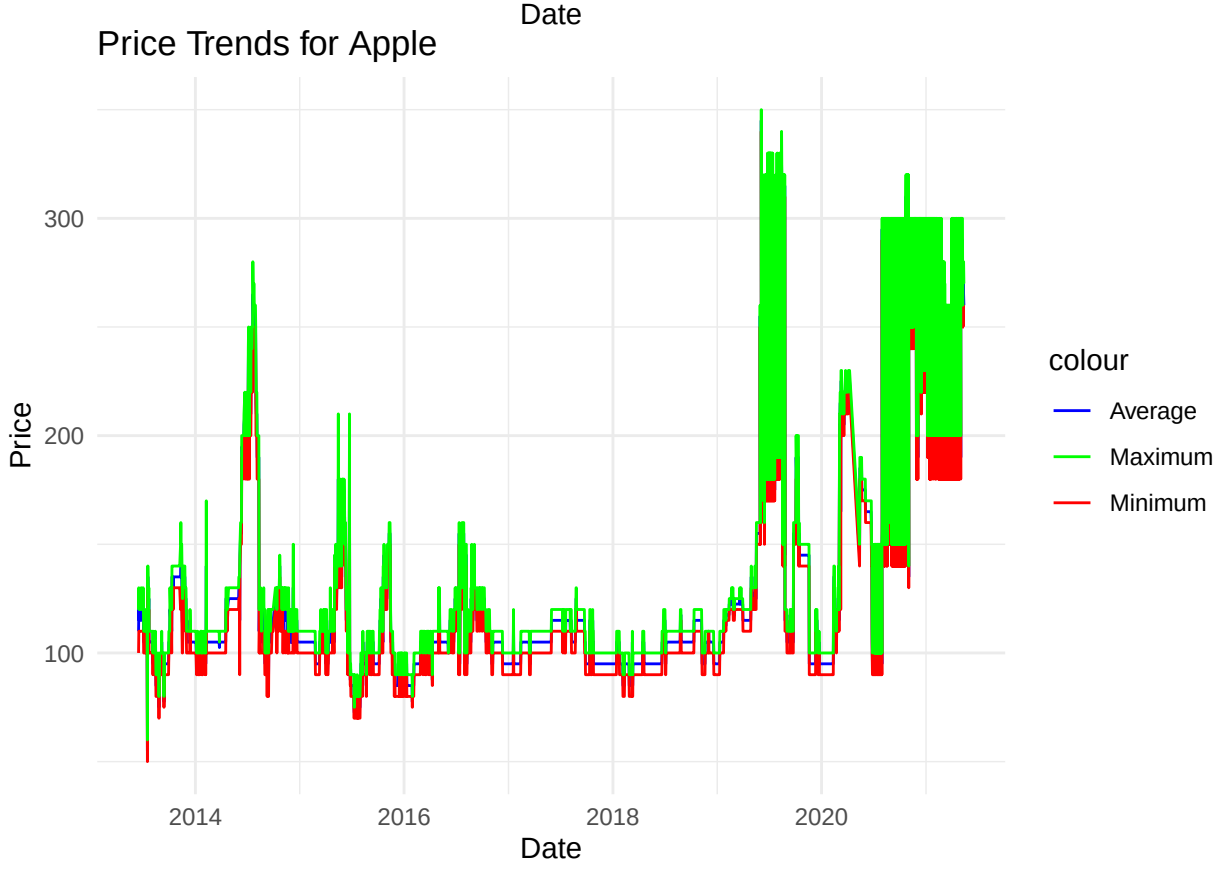

Price Trends for Potato Red
120
90
60
30
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

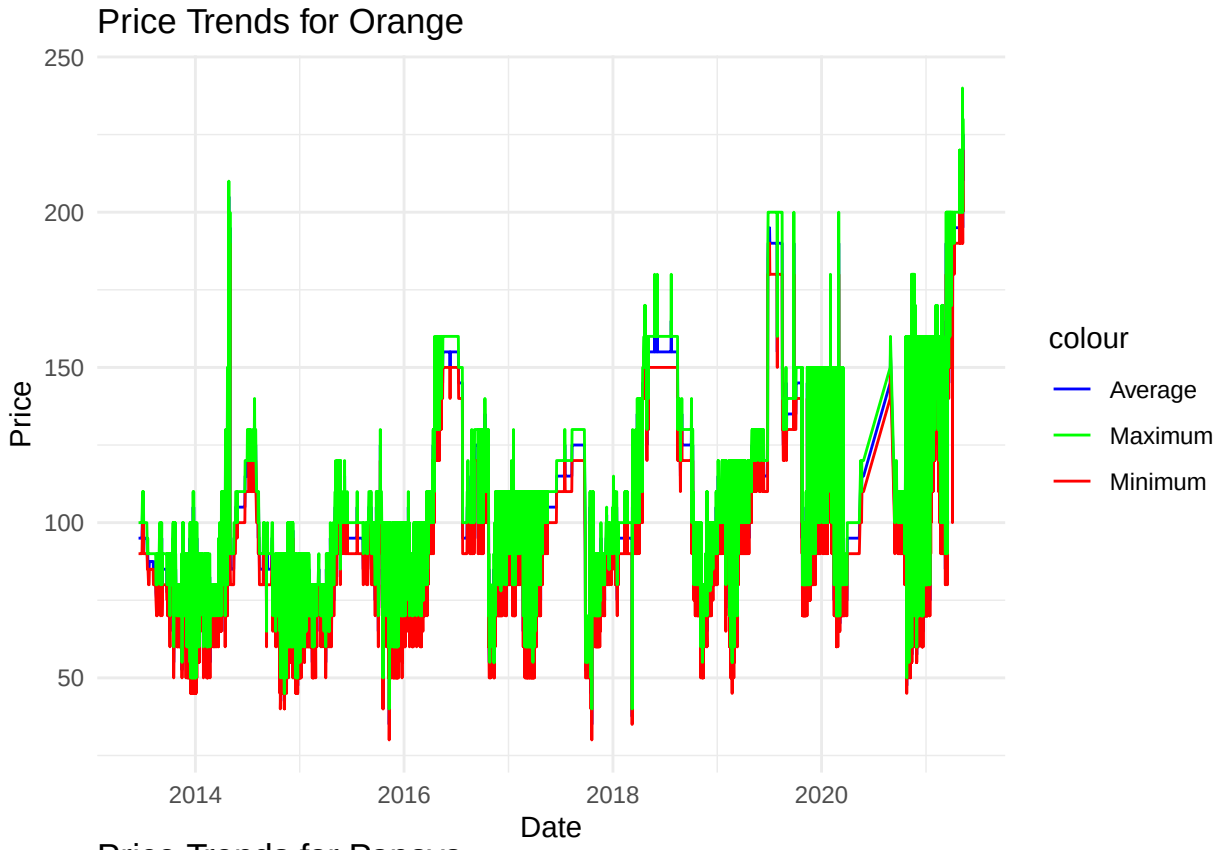

Price Trends for Potato White
100
75
50
25
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

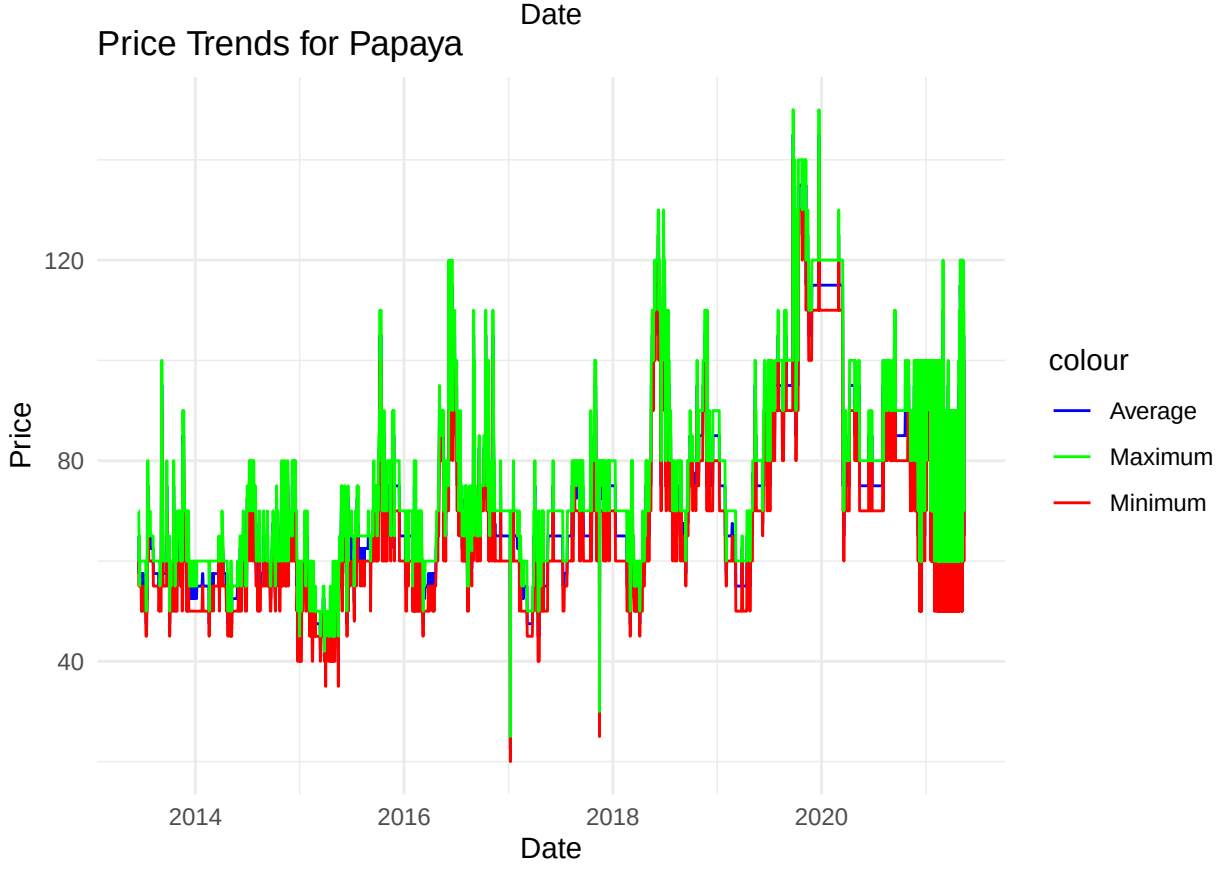

Price Trends for Onion Dry
200
150
100
50
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

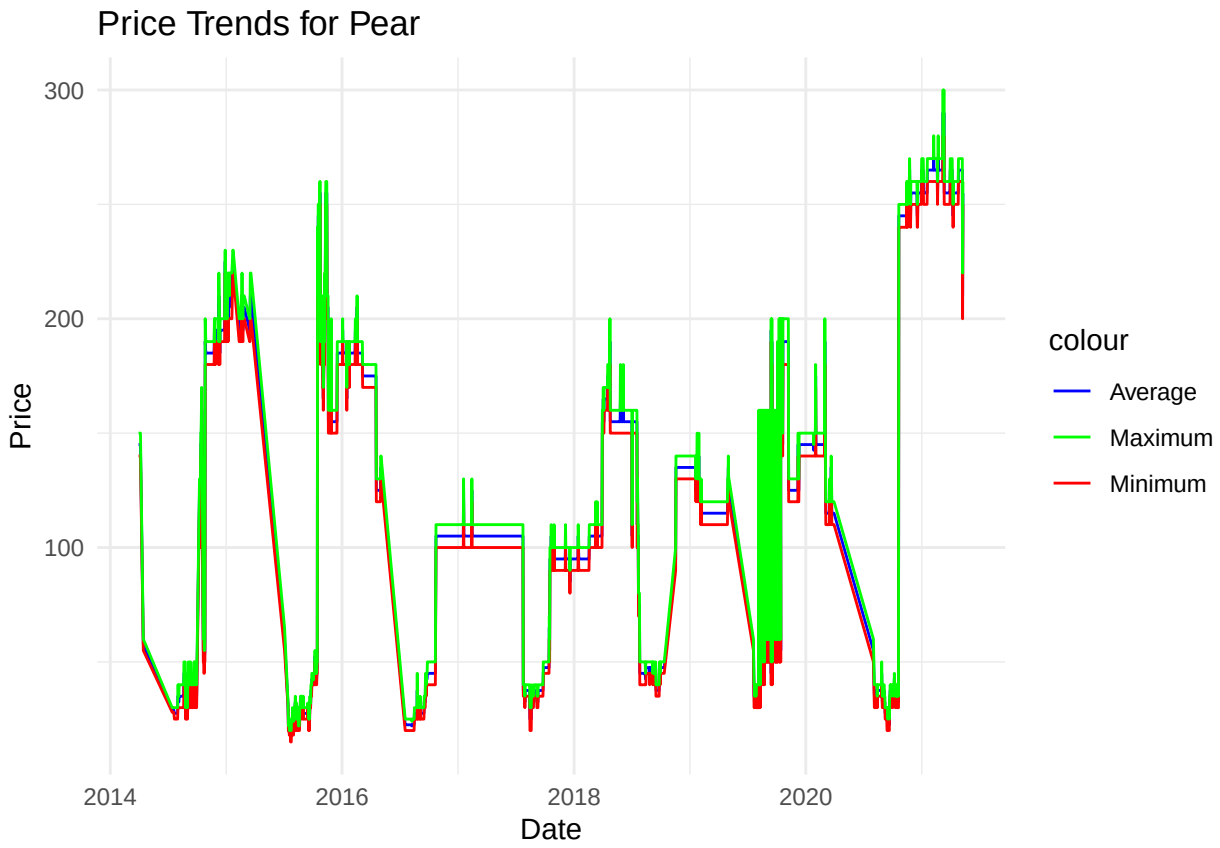

Price Trends for Other
2000
1500
1000
500
0
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum


Price Trends for Onion Green
300
200
100
0
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum


Price Trends for Apple
300
200
100
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

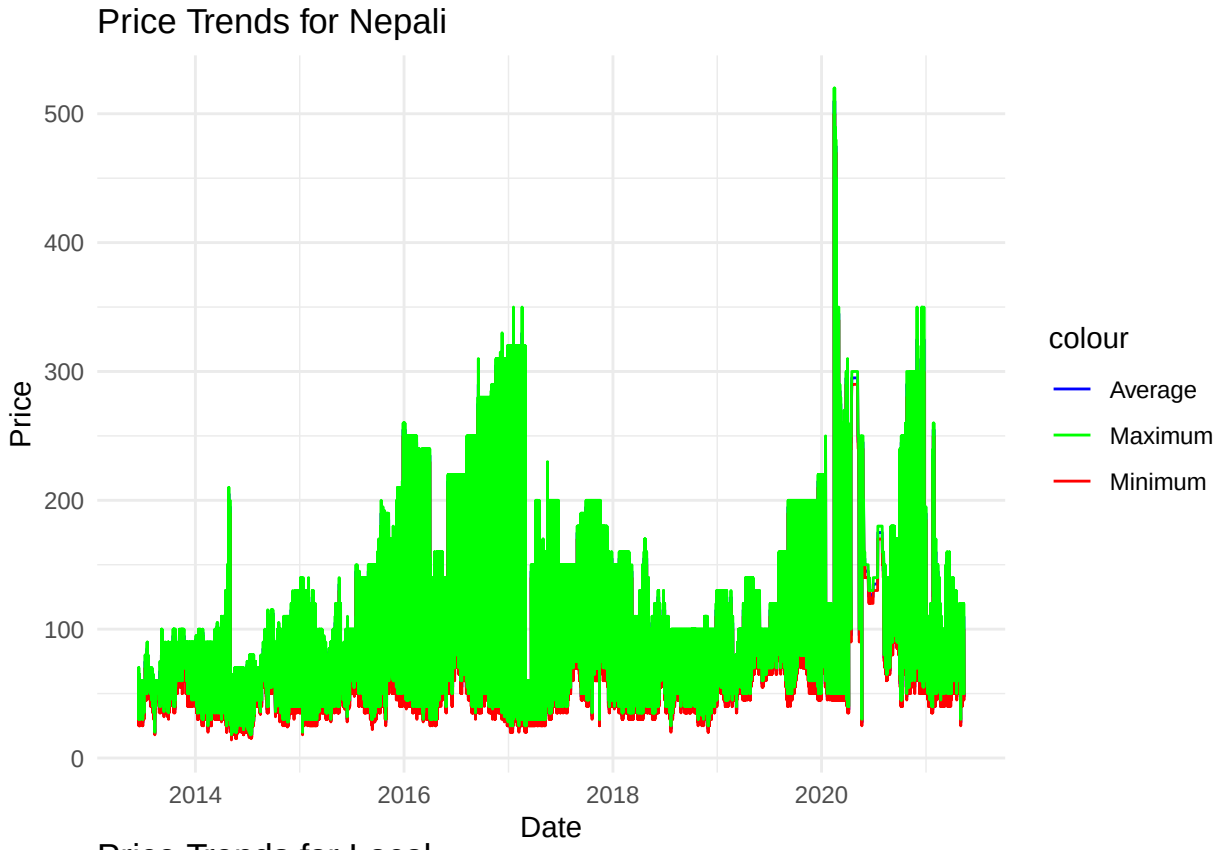

Price Trends for Orange
250
200
150
100
50
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

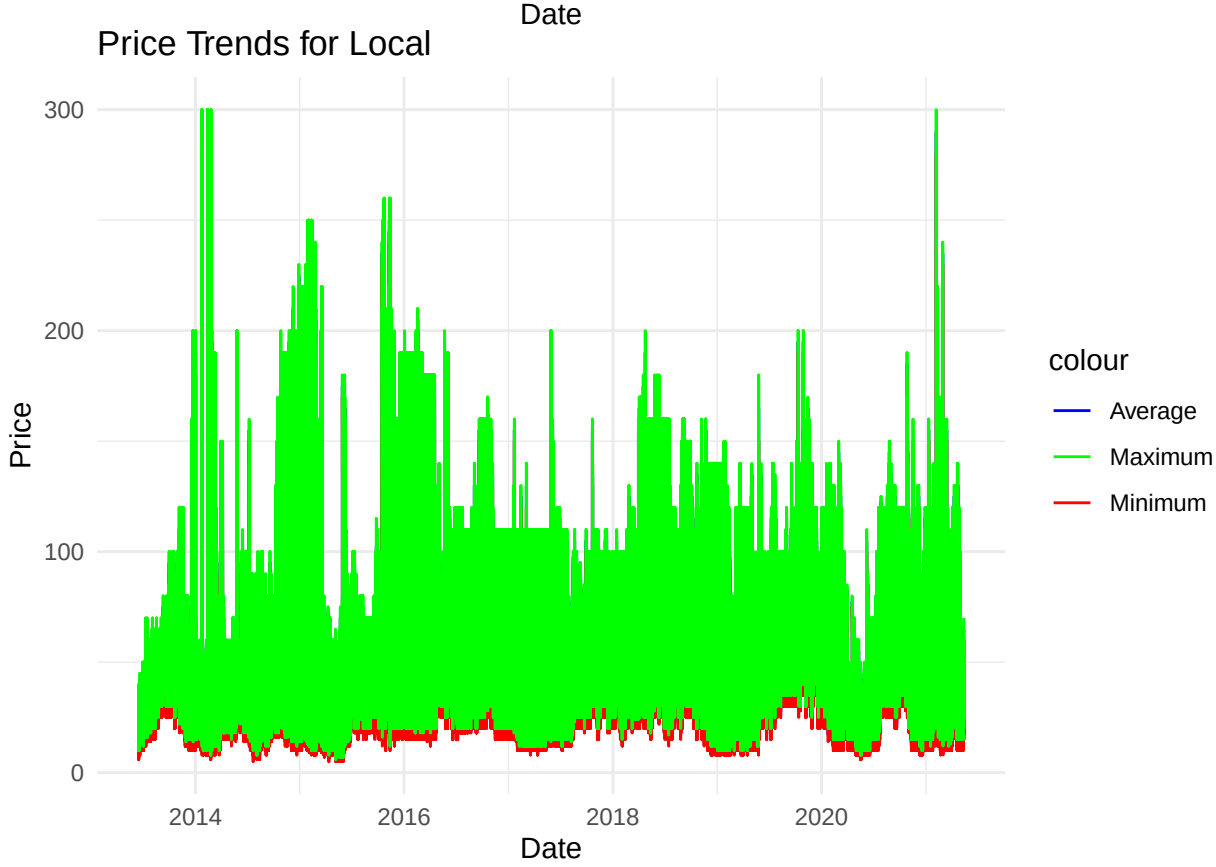

Price Trends for Papaya
120
80
40
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

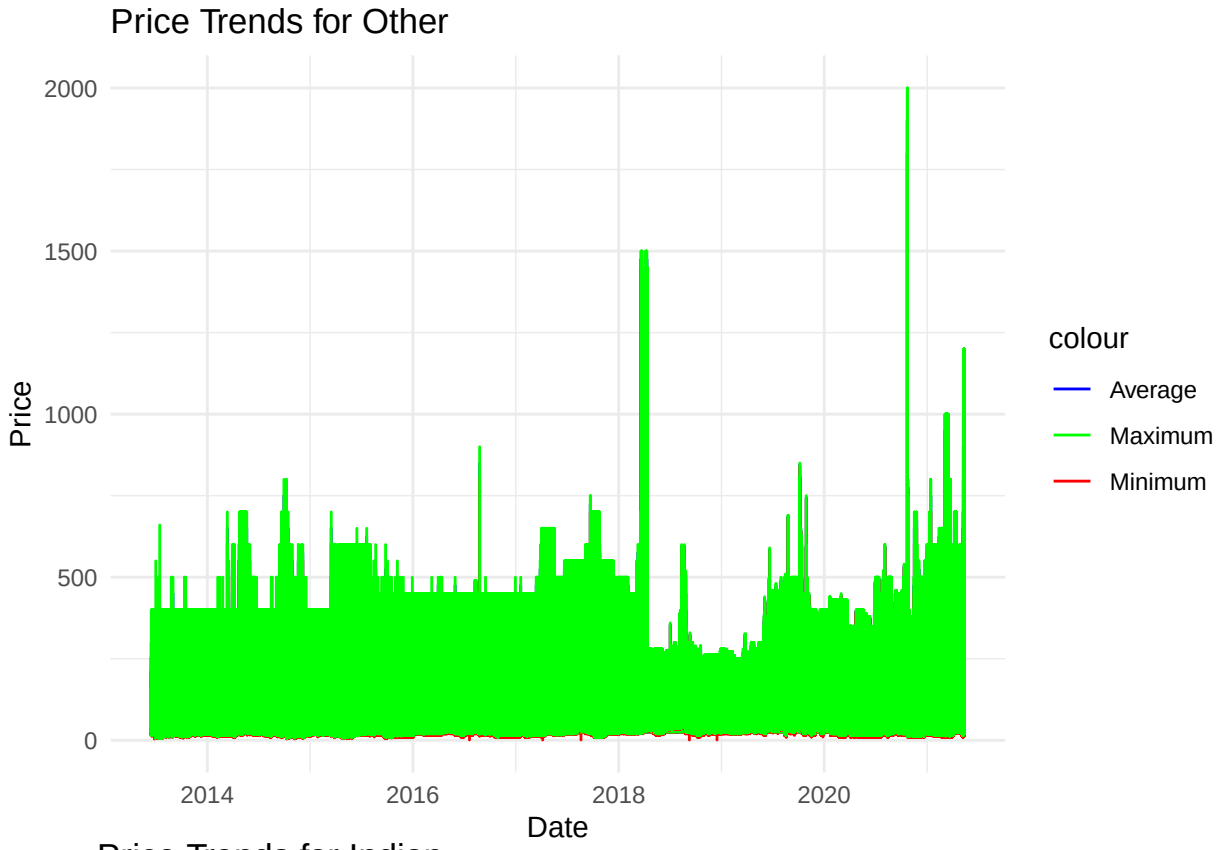


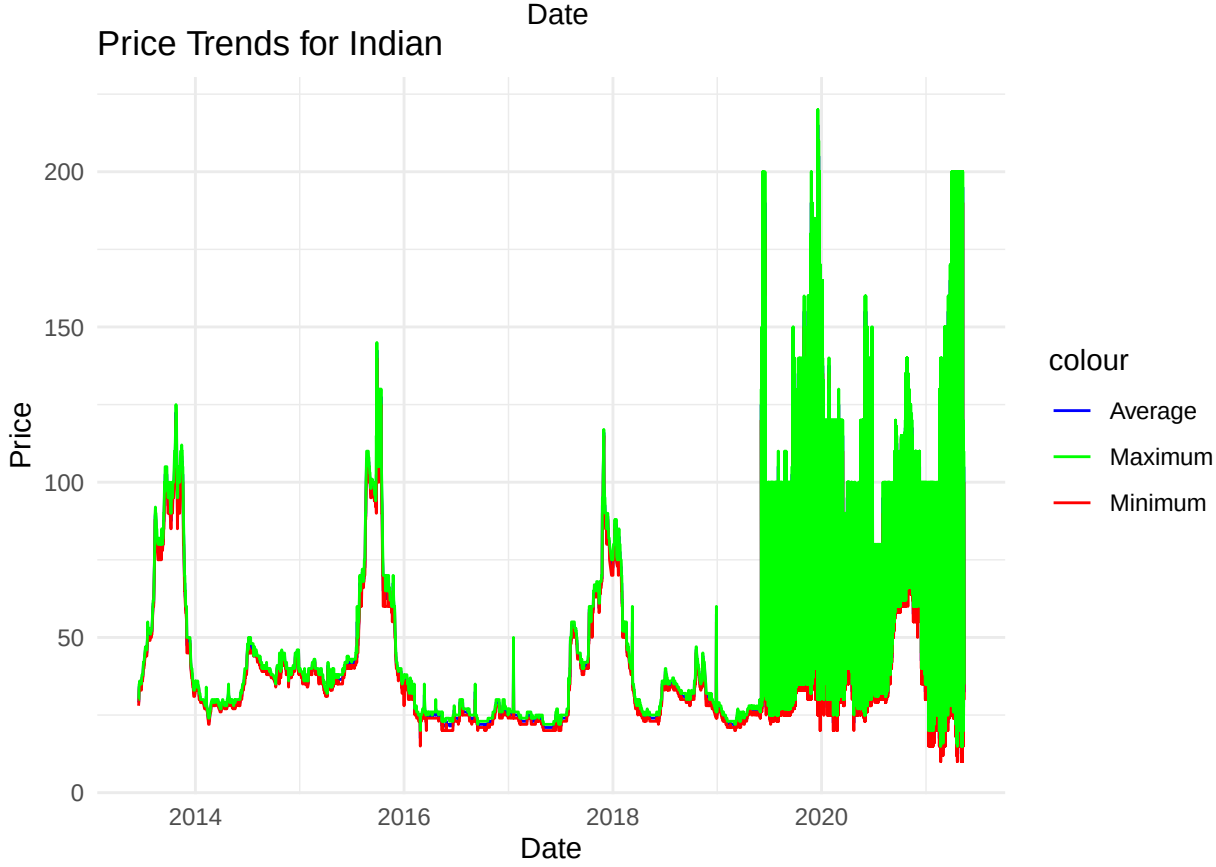
Plot Charts for Varieties

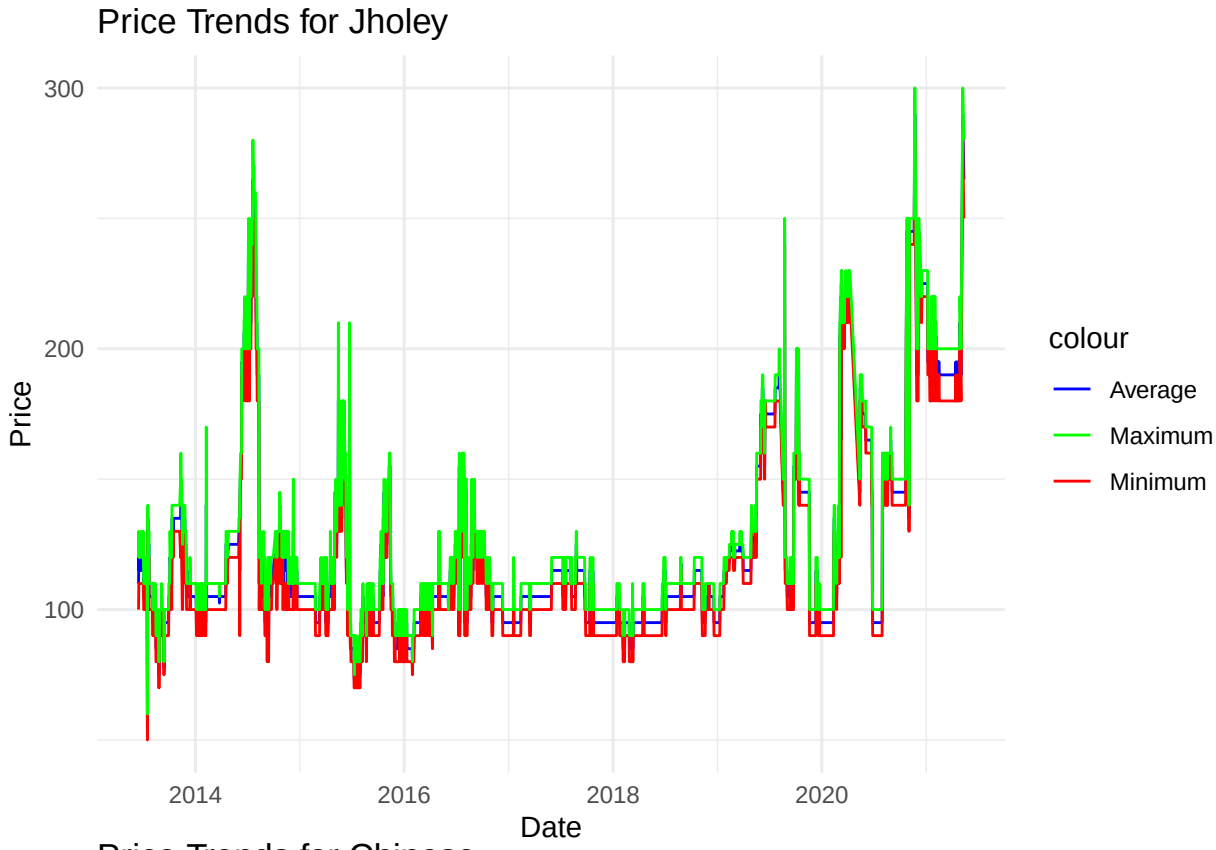
```
# Create plots for different varieties
create_variety_plot <- function(variety_name) {
  data_filtered <- data %>% filter(Variety == variety_name)

  ggplot(data_filtered, aes(x = Date)) +
    geom_line(aes(y = Average, color = "Average")) +
    geom_line(aes(y = Minimum, color = "Minimum")) +
    geom_line(aes(y = Maximum, color = "Maximum")) +
    labs(title = paste("Price Trends for", variety_name),
         x = "Date",
         y = "Price") +
    scale_color_manual(values = c("Average" = "blue", "Minimum" = "red", "Maximum" = "green")) +
    theme_minimal()
}

unique_variety <- unique(data$Variety)

for (variety in unique_variety) {
  plot <- create_variety_plot(variety)
  print(plot)
}
```

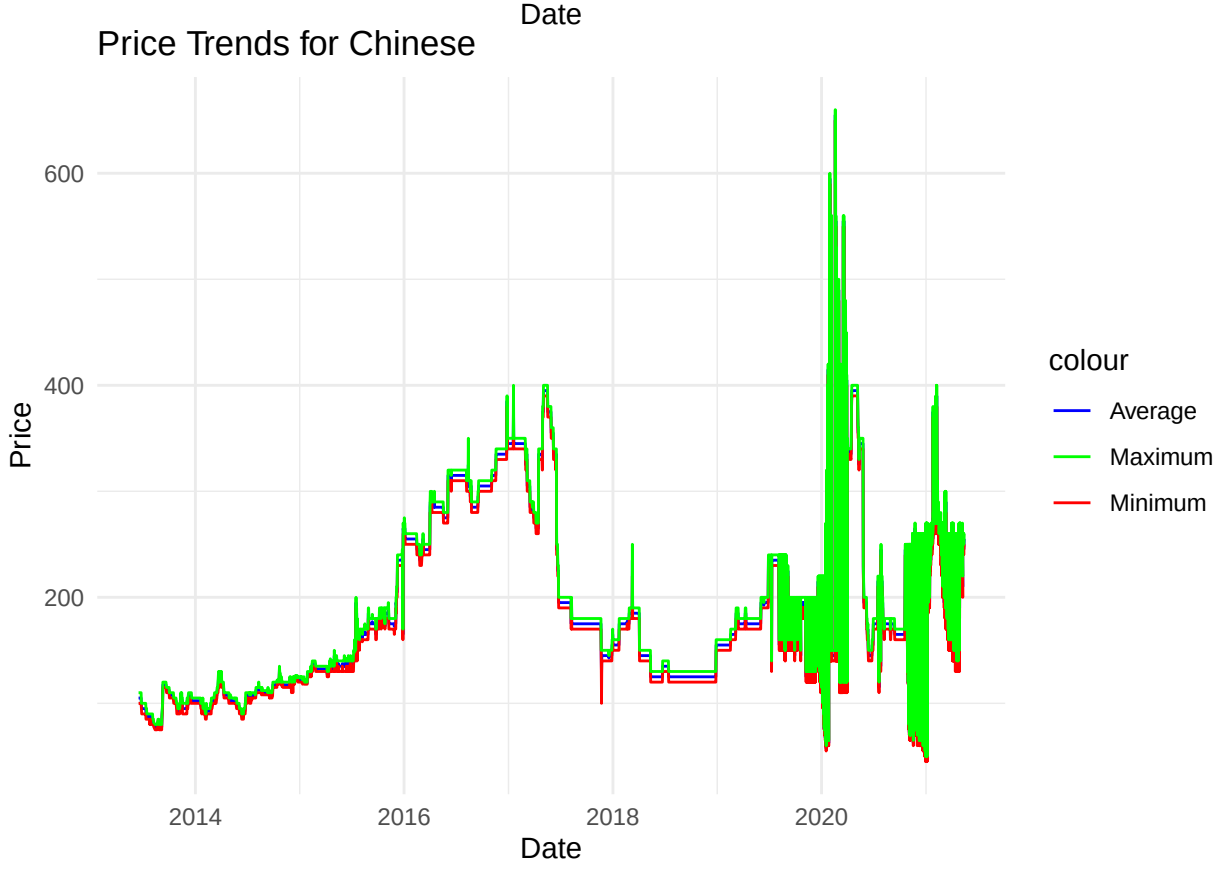

Price Trends for Nepali
Price
500
400
300
200
100
0
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

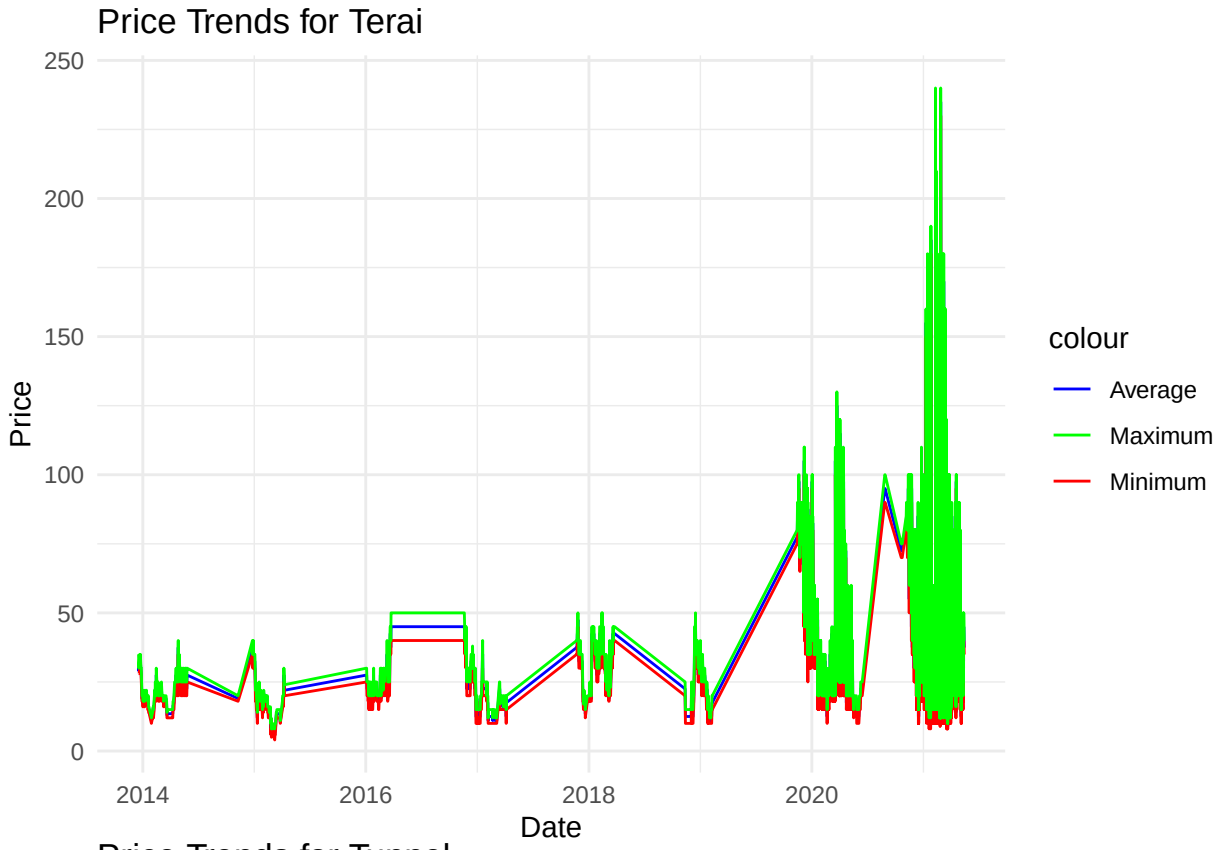

Price Trends for Local
Price
300
200
100
0
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

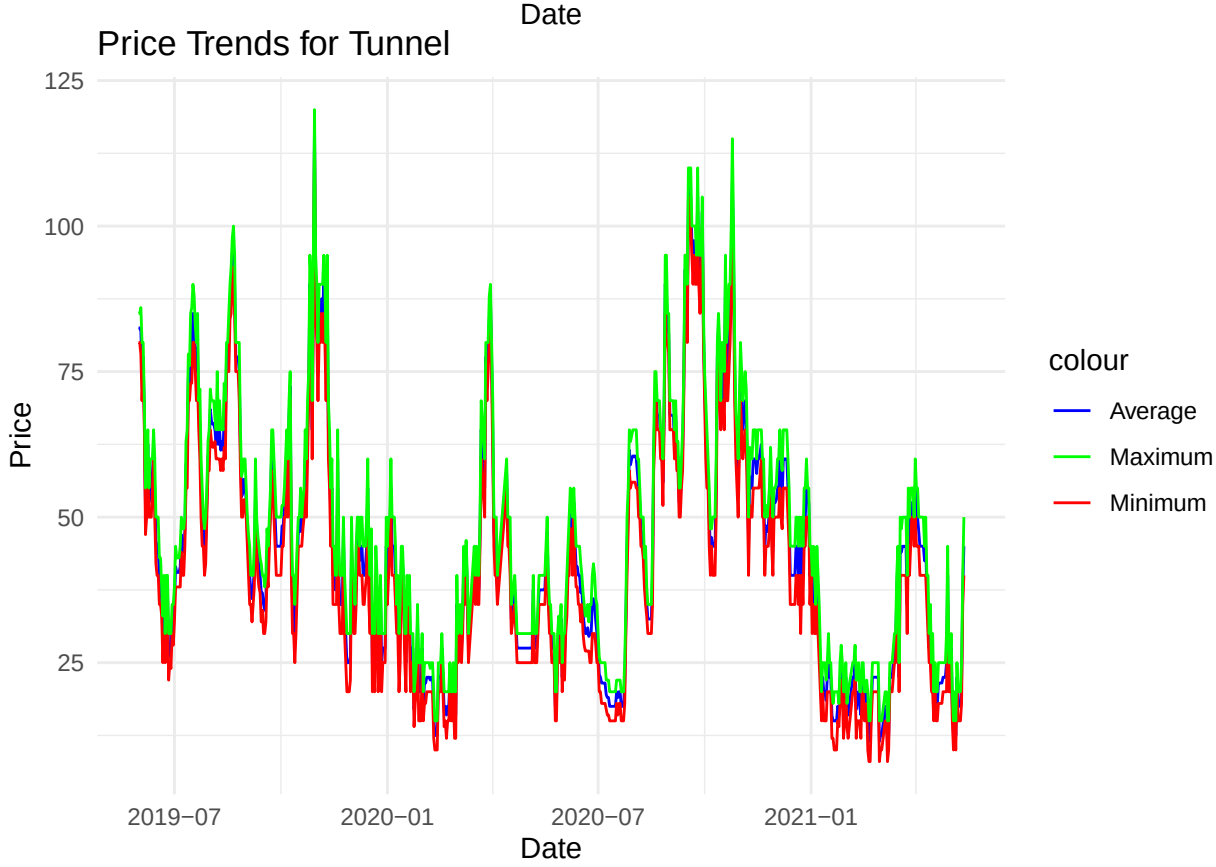

Price Trends for Other
2000
1500
1000
500
0
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

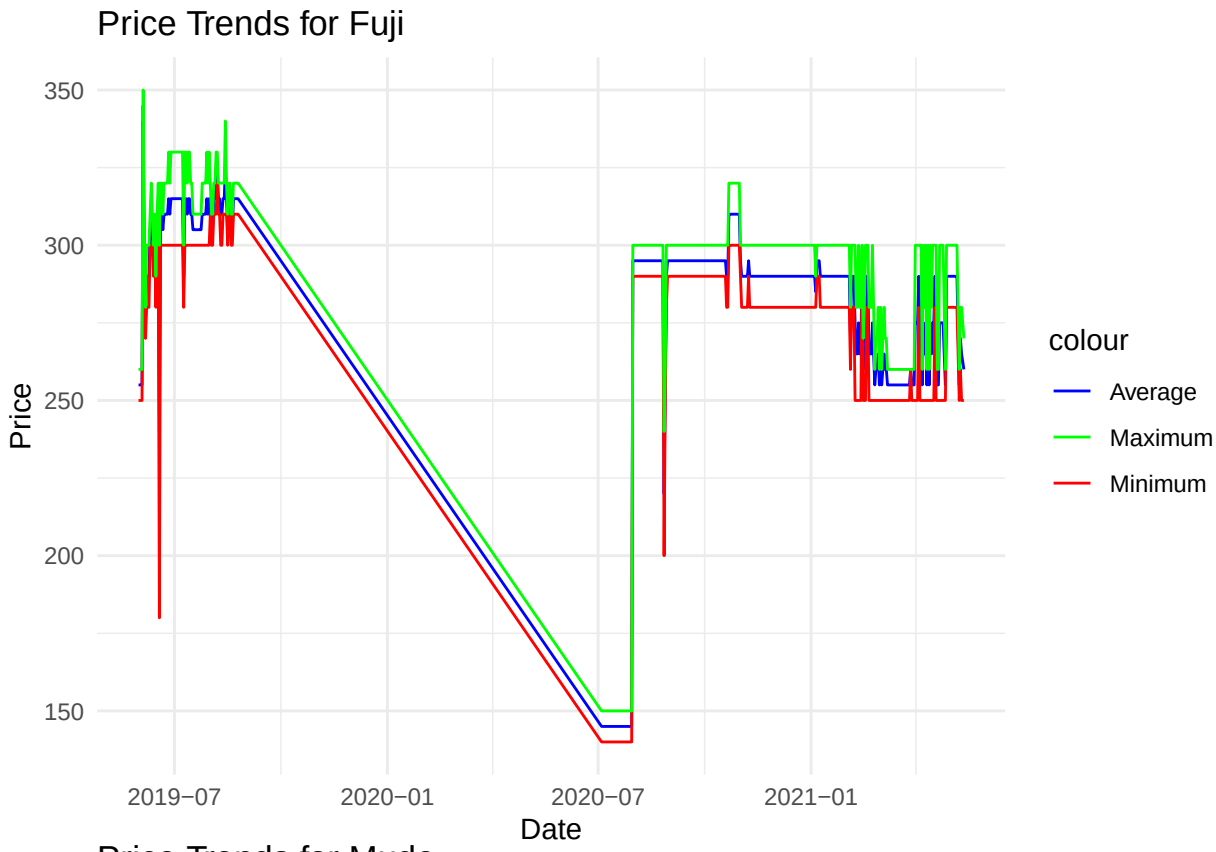

Price Trends for Indian
200
150
100
50
0
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum


Price Trends for Jholey
300
200
100
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum


Price Trends for Chinese
600
400
200
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum


Price Trends for Terai
250
200
150
100
50
0
Price
2014
2016
2018
2020
Date
colour
Average
Maximum
Minimum

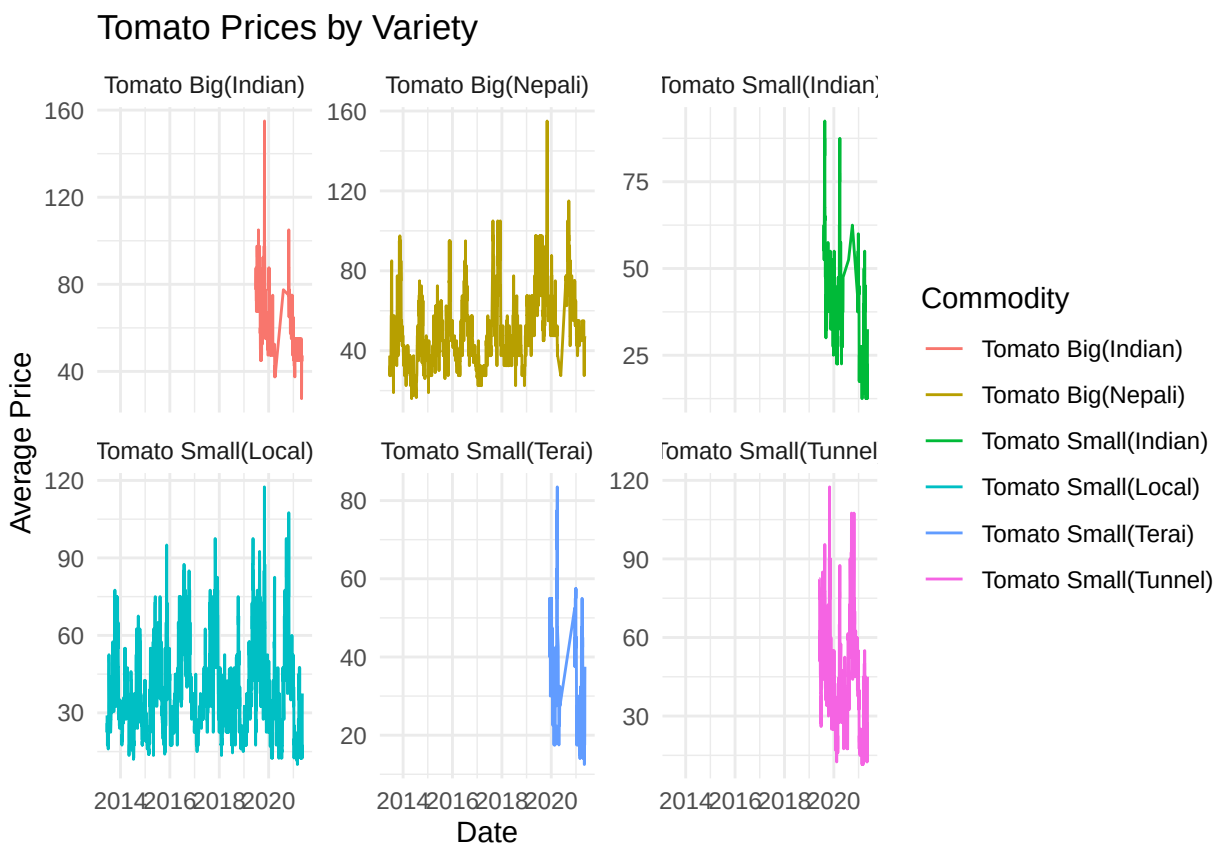

Price Trends for Tunnel
125
100
75
50
25
Price
2019−07
2020−01
2020−07
2021−01
Date
colour
Average
Maximum
Minimum



Tomato Data Analysis

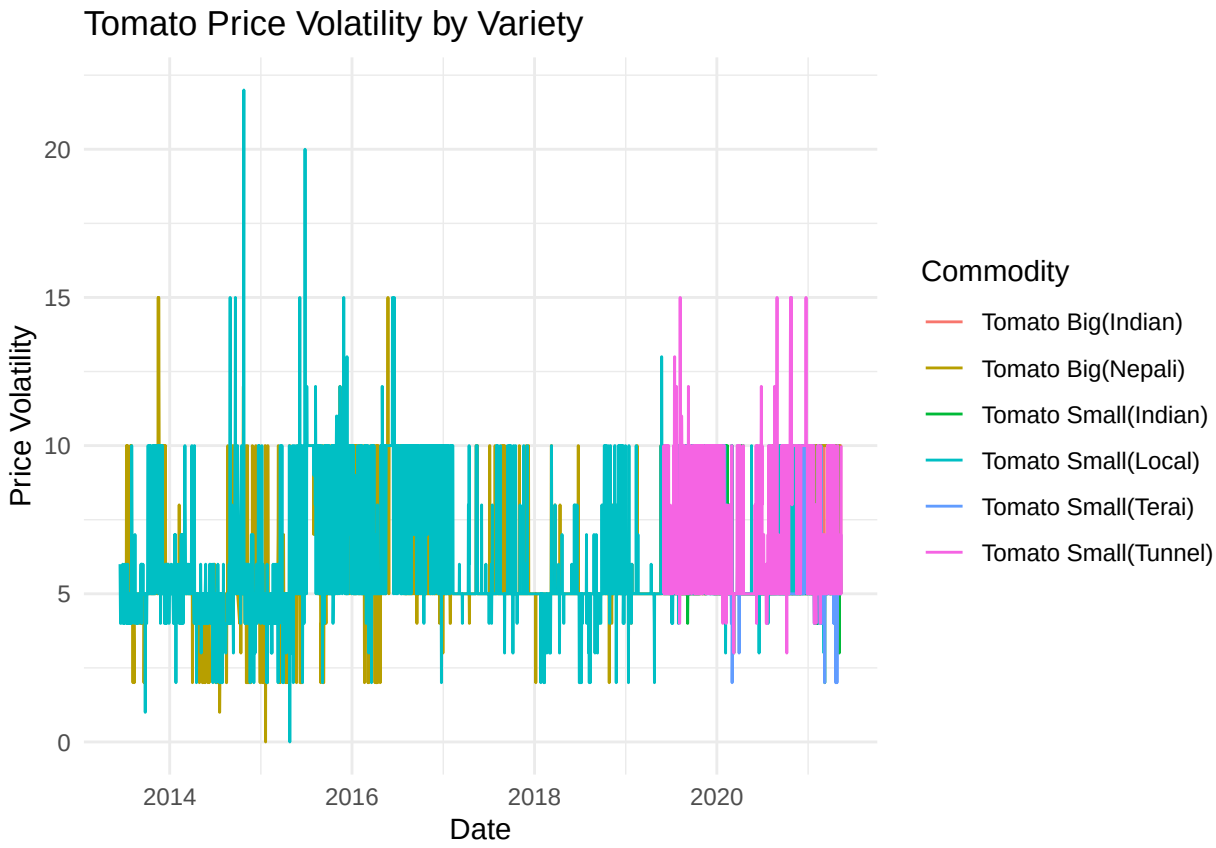
Plot Tomato Varieties

```
# Analysis for tomato data

tomato_data <- data %>% filter(grepl("Tomato", Commodity))

variety_plot <- ggplot(tomato_data, aes(x = Date, y = Average, color = Commodity)) +
  geom_line() +
  facet_wrap(~ Commodity, scales = "free_y") +
  labs(title = "Tomato Prices by Variety", x = "Date", y = "Average Price") +
  theme_minimal()

print(variety_plot)
```



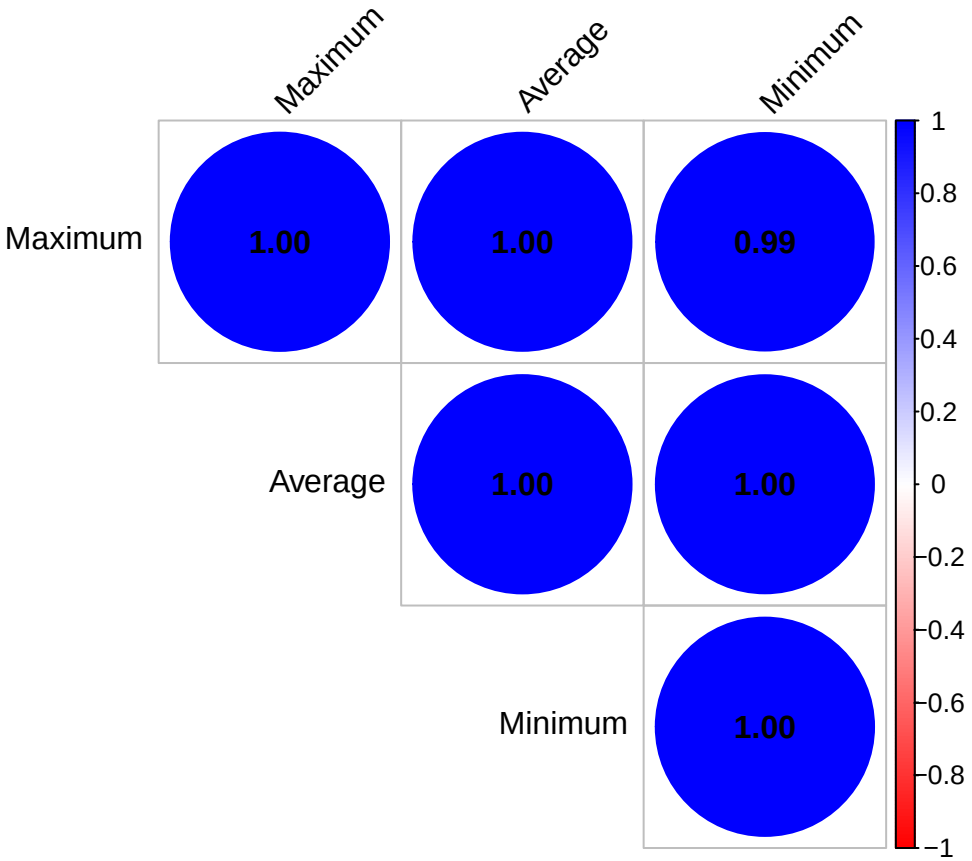
Volatility

```
tomato_data$Volatility <- tomato_data$Maximum - tomato_data$Minimum

volatility_variety <- ggplot(tomato_data, aes(x = Date, y = Volatility, color = Commodity)) +
  geom_line() +
  labs(title = "Tomato Price Volatility by Variety", x = "Date", y = "Price Volatility") +
  theme_minimal()

print(volatility_variety)
```



Price Corrplot

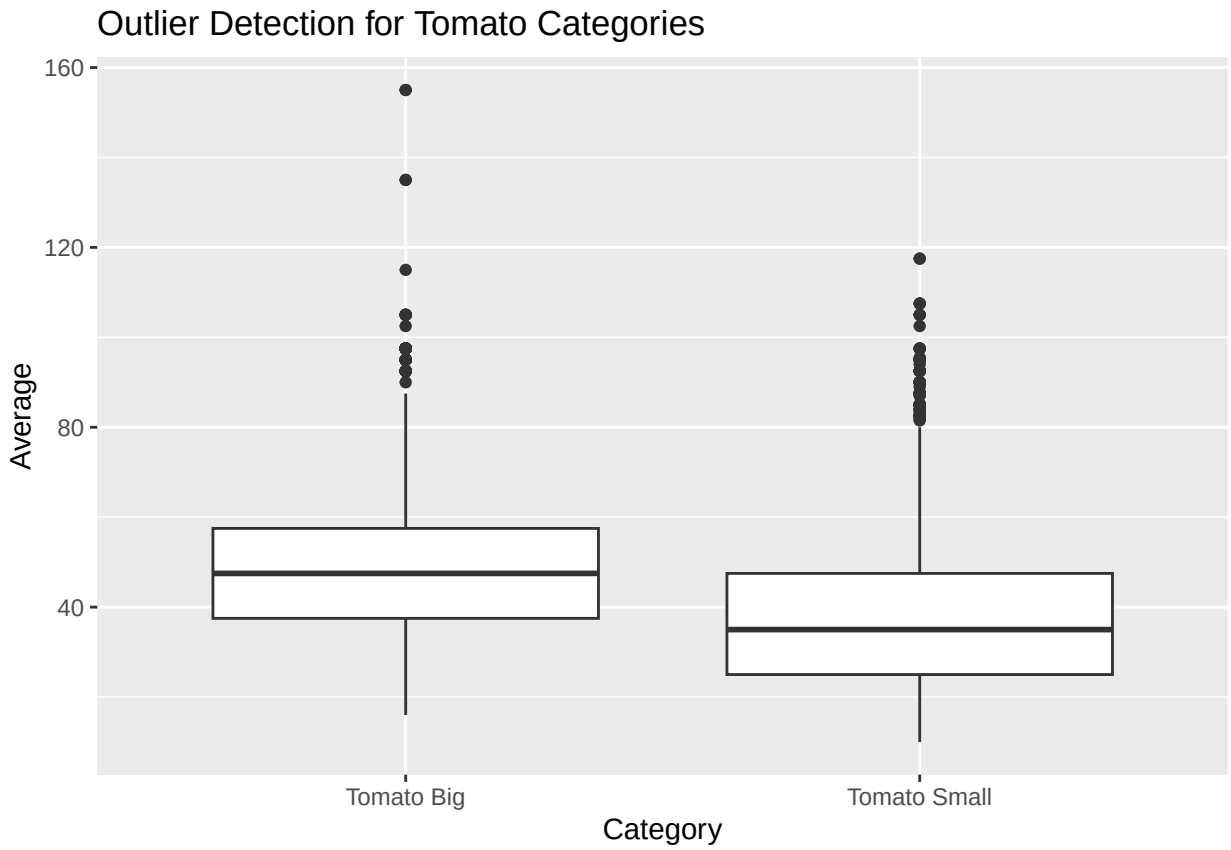
```
tomato_data <- as.data.frame(tomato_data)

tomato_data$Average <- as.numeric(tomato_data$Average)
tomato_data$Minimum <- as.numeric(tomato_data$Minimum)
tomato_data$Maximum <- as.numeric(tomato_data$Maximum)

# Select numeric columns (Minimum, Maximum, Average)
selected_data <- tomato_data[, c("Minimum", "Maximum", "Average")]

# Calculate correlation matrix
price_cor <- cor(selected_data, use = "pairwise.complete.obs")

corrplot(price_cor,
         method = "circle",        # Use circles for correlation values
         type = "upper",           # Show only the upper triangle
         order = "AOE",            # Alphabetical order for labels
         tl.col = "black",         # Black text labels
         tl.srt = 45,              # Rotate text labels
         addCoef.col = "black",    # Add coefficients on the plot
         col = colorRampPalette(c("red", "white", "blue"))(200))  # Custom color scale
```



Outlier Detection

Tomato Big (Nepali) and Tomato Small (Tunnel) have the most significant outliers. Prices outliers in Tomato Big (Nepali) cluster around 105, 135, and 155 while Tomato Small (Tunnel) is consistently around 105. These trends are likely due to market disruptions such as demand spikes and supply shortages and niche production practices such as tunnel farming.

```
# Detect outliers using boxplot
outlier_plot <- ggplot(tomato_data, aes(y = Average, x = Category)) +
  geom_boxplot() +
  labs(title = "Outlier Detection for Tomato Categories")

print(outlier_plot)
```




```
# Z-score method
tomato_data$z_score <- scale(tomato_data$Average)
outliers <- tomato_data[abs(tomato_data$z_score) > 3, ]
print(outliers)
```

```
##          SN            Commodity       Date Unit Minimum Maximum Average
## 2800  99815   Tomato Big(Nepali) 2017-08-22   Kg     100     110   105.0
## 2940 104395   Tomato Big(Nepali) 2017-11-01   Kg     100     110   105.0
## 3012 106761   Tomato Big(Nepali) 2017-12-08   Kg     100     110   105.0
## 3014 106825   Tomato Big(Nepali) 2017-12-09   Kg     100     105   102.5
## 4287 146043   Tomato Big(Indian) 2019-08-03   Kg     100     110   105.0
## 4690 152318  Tomato Small(Local) 2019-10-30   Kg     115     120   117.5
## 4691 152319 Tomato Small(Tunnel) 2019-10-30   Kg     115     120   117.5
## 4692 152375   Tomato Big(Nepali) 2019-10-31   Kg     130     140   135.0
## 4693 152376   Tomato Big(Indian) 2019-10-31   Kg     130     140   135.0
## 4696 152440   Tomato Big(Nepali) 2019-11-01   Kg     150     160   155.0
## 4697 152441   Tomato Big(Indian) 2019-11-01   Kg     150     160   155.0
## 5906 173866   Tomato Big(Nepali) 2020-08-31   Kg     100     110   105.0
## 5957 175072   Tomato Big(Nepali) 2020-09-17   Kg     110     120   115.0
## 5959 175074 Tomato Small(Tunnel) 2020-09-17   Kg     100     110   105.0
## 5961 175146 Tomato Small(Tunnel) 2020-09-18   Kg     105     110   107.5
## 5964 175215 Tomato Small(Tunnel) 2020-09-19   Kg     100     110   105.0
## 5982 175638 Tomato Small(Tunnel) 2020-09-25   Kg     100     110   105.0
## 5994 175928 Tomato Small(Tunnel) 2020-09-29   Kg     100     105   102.5
## 6071 177795   Tomato Big(Indian) 2020-10-25   Kg     100     110   105.0
## 6072 177796  Tomato Small(Local) 2020-10-25   Kg     100     115   107.5
## 6073 177797 Tomato Small(Tunnel) 2020-10-25   Kg     100     115   107.5
```

```
##          Category Variety Volatility  z_score
## 2800   Tomato Big  Nepali         10 3.277806
## 2940   Tomato Big  Nepali         10 3.277806
## 3012   Tomato Big  Nepali         10 3.277806
## 3014   Tomato Big  Nepali          5 3.144043
## 4287   Tomato Big  Indian         10 3.277806
## 4690 Tomato Small   Local          5 3.946621
## 4691 Tomato Small  Tunnel          5 3.946621
## 4692   Tomato Big  Nepali         10 4.882961
## 4693   Tomato Big  Indian         10 4.882961
## 4696   Tomato Big  Nepali         10 5.953065
## 4697   Tomato Big  Indian         10 5.953065
## 5906   Tomato Big  Nepali         10 3.277806
## 5957   Tomato Big  Nepali         10 3.812858
## 5959 Tomato Small  Tunnel         10 3.277806
## 5961 Tomato Small  Tunnel          5 3.411569
## 5964 Tomato Small  Tunnel         10 3.277806
## 5982 Tomato Small  Tunnel         10 3.277806
## 5994 Tomato Small  Tunnel          5 3.144043
## 6071   Tomato Big  Indian         10 3.277806
## 6072 Tomato Small   Local         15 3.411569
## 6073 Tomato Small  Tunnel         15 3.411569
```

Principal Component Analysis

```
#PCA for tomato category
tomato_numeric <- tomato_data[, c("Minimum", "Maximum", "Average", "Volatility")]

# standardized data for pca
tomato_scaled <- scale(tomato_numeric)
# perform pca
pca <- prcomp(tomato_scaled, center = TRUE, scale. = TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2       PC3       PC4
## Standard deviation     1.7588 0.9522 2.353e-14 1.739e-15
## Proportion of Variance 0.7733 0.2266 0.000e+00 0.000e+00
## Cumulative Proportion  0.7733 1.0000 1.000e+00 1.000e+00
```

```
# Print PCA Loadings (Rotation)
cat("\nPrincipal Component Loadings:\n")
```

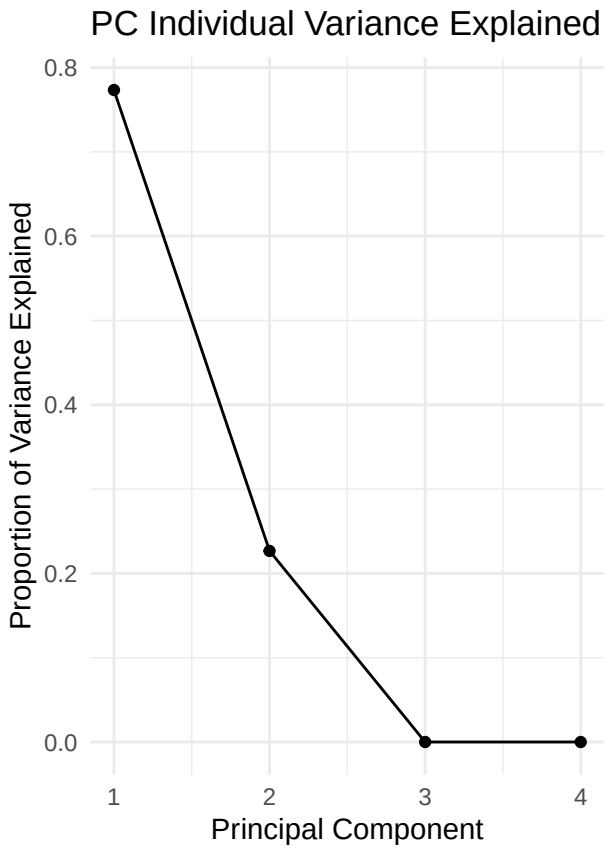
```
##
## Principal Component Loadings:
```

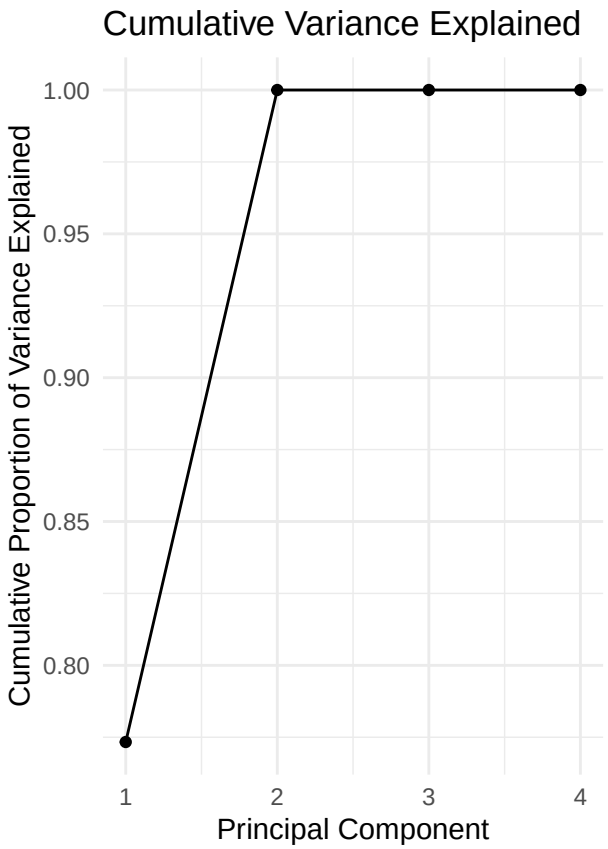
```
print(pca$rotation)
```

```
##                  PC1         PC2         PC3         PC4
## Minimum    0.5596096 -0.18570116 -0.26336058 -0.76354001
## Maximum    0.5675875 -0.06165696 -0.54096251  0.61757785
## Average    0.5646629 -0.12288251  0.79858291  0.16828847
## Volatility 0.2140964  0.97294059  0.01631282 -0.08534189
```

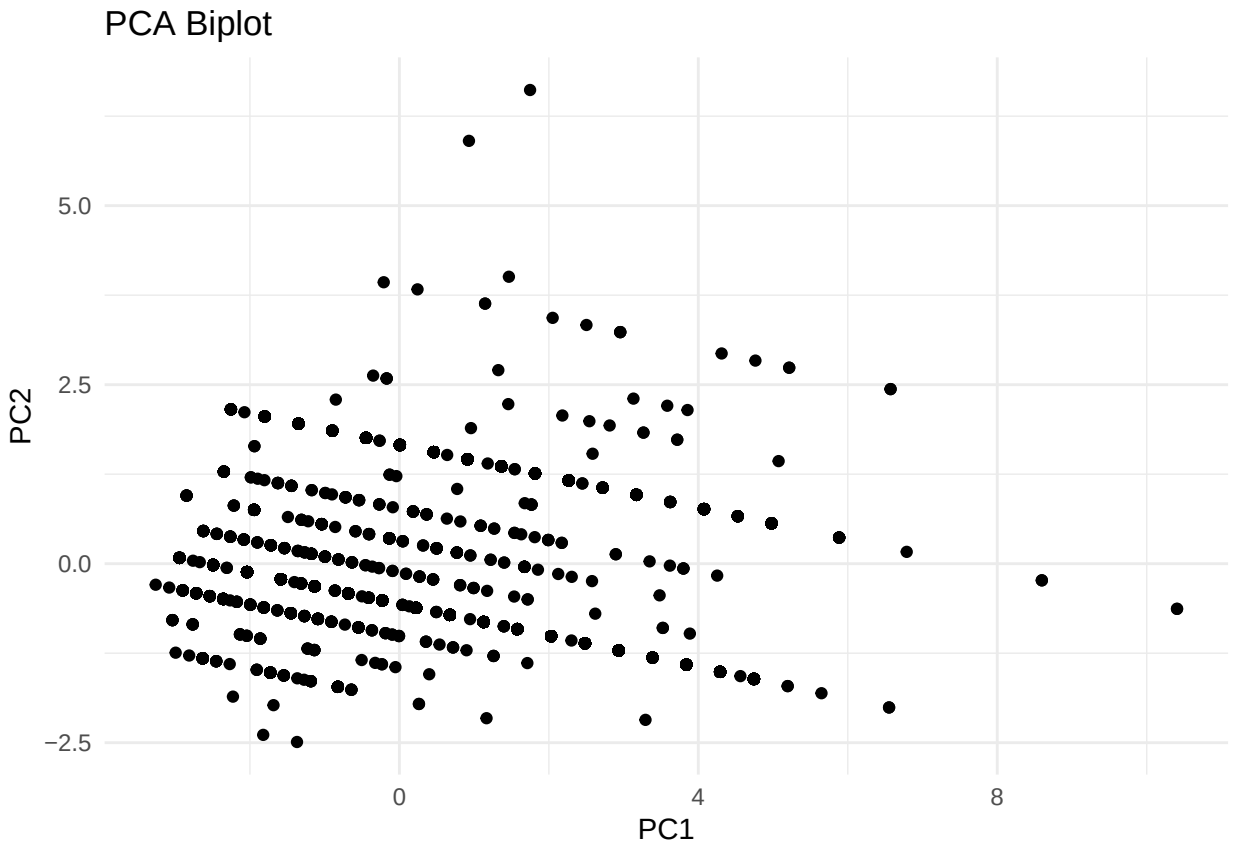
```
# Variance explained by each principal component
pca_variance <- pca$sdev^2
```

```r
proportion_variance <- pca_variance / sum(pca_variance)
cumulative_variance <- cumsum(proportion_variance)

# Create a dataframe for plotting
pca_df <- data.frame(
  Principal_Component = 1:length(proportion_variance),
  Proportion_Variance = proportion_variance,
  Cumulative_Variance = cumulative_variance
)

# Plot Individual Variance Explained
plot1 <- ggplot(pca_df, aes(x = Principal_Component, y = Proportion_Variance)) +
  geom_point() +
  geom_line() +
  labs(title = "PC Individual Variance Explained",
       x = "Principal Component",
       y = "Proportion of Variance Explained") +
  theme_minimal()

# Plot Cumulative Variance Explained
plot2 <- ggplot(pca_df, aes(x = Principal_Component, y = Cumulative_Variance)) +
  geom_point() +
  geom_line() +
  labs(title = "Cumulative Variance Explained",
       x = "Principal Component",
       y = "Cumulative Proportion of Variance Explained") +
  theme_minimal()

# Combine both plots into one window
library(gridExtra)  # For side-by-side plots
grid.arrange(plot1, plot2, ncol = 2)
```






```r
pca_scores <- data.frame(pca$x)
plot3 <- ggplot(pca_scores, aes(x = PC1, y = PC2)) +
  geom_point() +
  labs(title = "PCA Biplot", x = "PC1", y = "PC2") +
  theme_minimal()

print(plot3)
```



Model Selection ### Linear Regression Models

```
set.seed(123)  # Set a seed for reproducibility

# Split index for 80% train / 20% test
split_index <- floor(0.8 * nrow(tomato_data))

# Shuffle the rows (optional for linear models)
tomato_data <- tomato_data[sample(nrow(tomato_data)), ]

# Create train and test sets
train_data <- tomato_data[1:split_index, ]
test_data <- tomato_data[(split_index + 1):nrow(tomato_data), ]

# Check the sizes
cat("Train size:", nrow(train_data), "\n")
```

```
## Train size: 5640
```

```
cat("Test size:", nrow(test_data), "\n")
```

```
## Test size: 1411
```

```
# Convert Date to numeric for regression
train_data$Date_Num <- as.numeric(as.Date(train_data$Date))
test_data$Date_Num <- as.numeric(as.Date(test_data$Date))

# Fit the model
```

```r
lm_avg_category <- lm(Average ~ Date_Num + Category, data = train_data)

# Predict on the test set
test_data$Predicted_Avg <- predict(lm_avg_category, newdata = test_data)

# View model summary
summary(lm_avg_category)
```

Category Model

```
##
## Call:
## lm(formula = Average ~ Date_Num + Category, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.223 -13.174  -3.387   9.825 101.608
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -1.218e+01  4.867e+00  -2.502   0.0124 *
## Date_Num               3.603e-03  2.783e-04  12.945   <2e-16 ***
## CategoryTomato Small -1.292e+01  4.762e-01 -27.134   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.48 on 5637 degrees of freedom
## Multiple R-squared:  0.1264, Adjusted R-squared:  0.126
## F-statistic: 407.6 on 2 and 5637 DF,  p-value: < 2.2e-16
```

```r
# Calculate VIF
vif_values <- vif(lm_avg_category)
print(vif_values)
```

```
## Date_Num Category
## 1.024107 1.024107
```

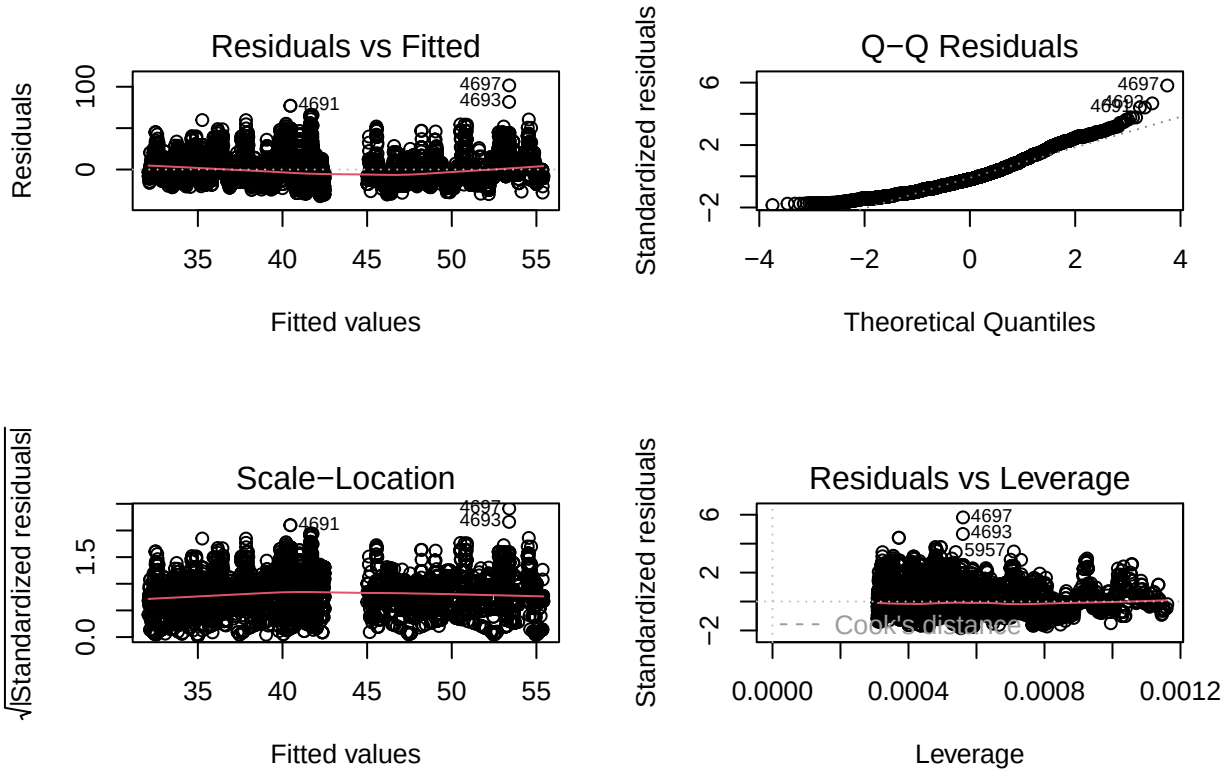
```r
# Fit a linear model
bp_test <- bptest(lm_avg_category)
print(bp_test)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_avg_category
## BP = 56.832, df = 2, p-value = 4.561e-13
```

```r
# Prepare data
x <- model.matrix(Average ~ Date_Num + Category, train_data)[, -1]
y <- train_data$Average

# Fit Lasso regression
lasso_model <- cv.glmnet(x, y, alpha = 1)  # Lasso
print(lasso_model$lambda.min)  # Optimal penalty
```

```
## [1] 0.02948584
```

```r
#plot
par(mfrow = c(2, 2))
plot(lm_avg_category)
```



Interaction Model

```r
# Fit the model with interaction
lm_avg_interaction <- lm(Average ~ Date_Num * Category, data = train_data)

# Predict on the test set
test_data$Predicted_Avg_Interaction <- predict(lm_avg_interaction, newdata = test_data)

# View model summary
summary(lm_avg_interaction)
```

```
##
## Call:
## lm(formula = Average ~ Date_Num * Category, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -34.249 -12.266  -3.638   8.838  97.902
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -9.739e+01  7.354e+00  -13.24   <2e-16 ***
## Date_Num                        8.488e-03  4.211e-04   20.16   <2e-16 ***
## CategoryTomato Small            1.348e+02  9.713e+00   13.88   <2e-16 ***
## Date_Num:CategoryTomato Small -8.417e-03  5.527e-04  -15.23   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.14 on 5636 degrees of freedom
## Multiple R-squared:  0.1609, Adjusted R-squared:  0.1604
## F-statistic: 360.2 on 3 and 5636 DF,  p-value: < 2.2e-16
```

```
# Calculate VIF
vif_values <- vif(lm_avg_interaction)
```

```
## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif
```

```
print(vif_values)
```

```
##          Date_Num          Category Date_Num:Category
##           2.44082         443.60407         452.68025
```

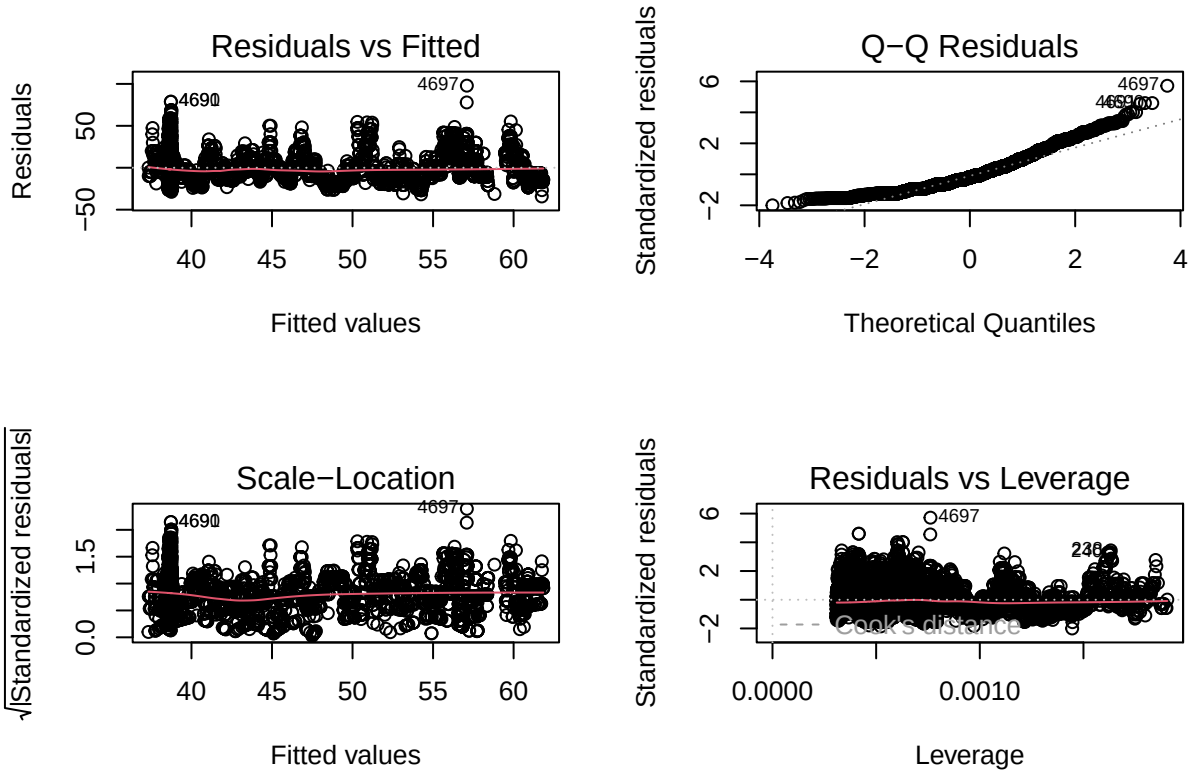
```
# Fit a linear model
bp_test <- bptest(lm_avg_interaction)
print(bp_test)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_avg_interaction
## BP = 54.371, df = 3, p-value = 9.35e-12
```

```
# Prepare data
x <- model.matrix(Average ~ Date_Num * Category, train_data)[, -1]
y <- train_data$Average

# Fit Lasso regression
lasso_model <- cv.glmnet(x, y, alpha = 1)  # Lasso
print(lasso_model$lambda.min)  # Optimal penalty
```

```
## [1] 0.0005924392
```

```
par(mfrow = c(2, 2))
plot(lm_avg_interaction)
```



Variety Model

```
# Convert Variety to factor
train_data$Variety <- as.factor(train_data$Variety)
test_data$Variety <- as.factor(test_data$Variety)

# Fit the model
lm_avg_variety <- lm(Average ~ Date_Num + Variety, data = train_data)

# Predict on the test set
test_data$Predicted_Avg_Variety <- predict(lm_avg_variety, newdata = test_data)

# View model summary
summary(lm_avg_variety)
```

```
##
## Call:
## lm(formula = Average ~ Date_Num + Variety, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -39.476 -12.589  -3.146   9.451 104.715
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -5.283e+00  6.244e+00  -0.846    0.397
## Date_Num       3.053e-03  3.372e-04   9.053  < 2e-16 ***
## VarietyLocal  -9.672e+00  8.622e-01 -11.218  < 2e-16 ***
## VarietyNepali  1.281e+00  8.830e-01   1.451    0.147
## VarietyTerai  -1.904e+01  1.377e+00 -13.833  < 2e-16 ***
## VarietyTunnel -5.972e+00  1.028e+00  -5.812 6.51e-09 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.68 on 5634 degrees of freedom
## Multiple R-squared:  0.1066, Adjusted R-squared:  0.1058
## F-statistic: 134.5 on 5 and 5634 DF,  p-value: < 2.2e-16
```

```
# Calculate VIF
vif_values <- vif(lm_avg_variety)
print(vif_values)
```

```
##              GVIF Df GVIF^(1/(2*Df))
## Date_Num 1.469659  1        1.212295
## Variety  1.469659  4        1.049306
```

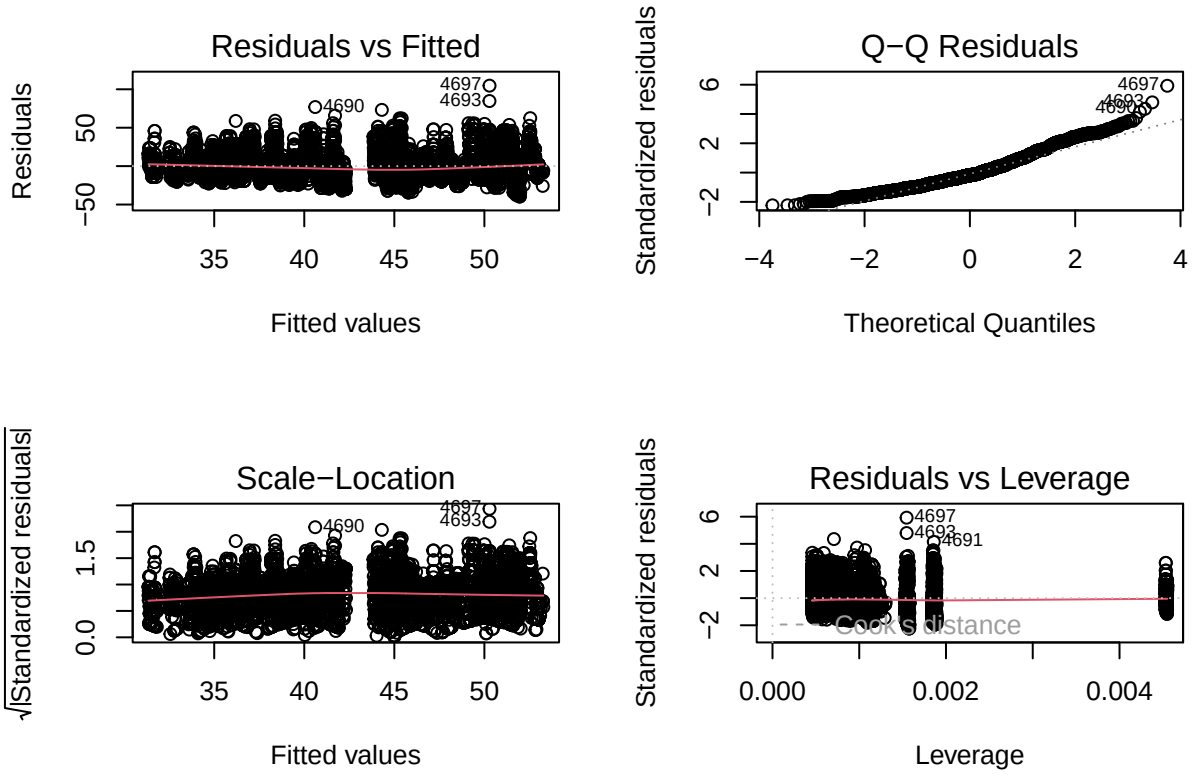
```
# Fit a linear model
bp_test <- bptest(lm_avg_variety)
print(bp_test)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_avg_variety
## BP = 149.16, df = 5, p-value < 2.2e-16
```

```
# Prepare data
x <- model.matrix(Average ~ Date_Num + Variety, train_data)[, -1]
y <- train_data$Average

# Fit Lasso regression
lasso_model <- cv.glmnet(x, y, alpha = 1)  # Lasso
print(lasso_model$lambda.min)  # Optimal penalty
```

```
## [1] 0.01572509
```

```
par(mfrow = c(2, 2))
plot(lm_avg_variety)
```



Combined Model

```
# Fit the combined model
lm_combined <- lm(Average ~ Date_Num + Category + Variety + Volatility, data = train_data)

# Predict on the test set
test_data$Predicted_Avg_Combined <- predict(lm_combined, newdata = test_data)

# View model summary
summary(lm_combined)
```

```
##
## Call:
## lm(formula = Average ~ Date_Num + Category + Variety + Volatility,
##     data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.069 -11.356  -2.906   9.129  88.709
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)           -4.516e+00  5.895e+00  -0.766  0.44367
## Date_Num               2.798e-03  3.168e-04   8.831  < 2e-16 ***
## CategoryTomato Small -2.282e+01  1.321e+00 -17.277  < 2e-16 ***
## VarietyLocal           4.470e+00  1.120e+00   3.992 6.63e-05 ***
## VarietyNepali         -6.895e+00  9.959e-01  -6.923 4.90e-12 ***
## VarietyTerai          -4.425e+00  1.504e+00  -2.942  0.00328 **
## VarietyTunnel          6.263e+00  1.239e+00   5.055 4.43e-07 ***
## Volatility             1.988e+00  9.713e-02  20.471  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.59 on 5632 degrees of freedom
## Multiple R-squared:  0.2135, Adjusted R-squared:  0.2125
## F-statistic: 218.4 on 7 and 5632 DF,  p-value: < 2.2e-16
```

```
# Calculate VIF
vif_values <- vif(lm_combined)
print(vif_values)
```

```
##                 GVIF Df GVIF^(1/(2*Df))
## Date_Num    1.472977  1        1.213663
## Category    8.743701  1        2.956975
## Variety    12.775032  4        1.374976
## Volatility  1.038659  1        1.019146
```

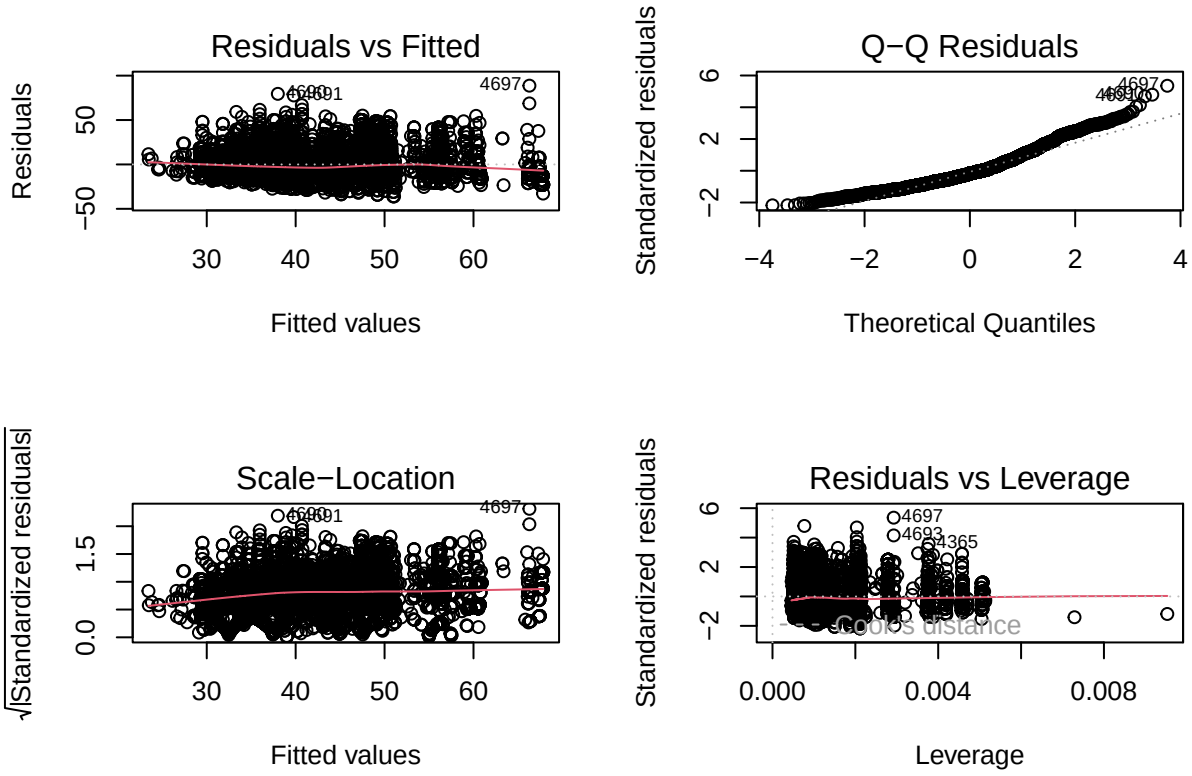
```
# Fit a linear model
bp_test <- bptest(lm_combined)
print(bp_test)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm_combined
## BP = 200.46, df = 7, p-value < 2.2e-16
```

```
# Prepare data
x <- model.matrix(Average ~ Date_Num + Category + Variety + Volatility, train_data)[, -1]
y <- train_data$Average

# Fit Lasso regression
lasso_model <- cv.glmnet(x, y, alpha = 1)  # Lasso
print(lasso_model$lambda.min)  # Optimal penalty
```

```
## [1] 0.008797535
```

```
par(mfrow = c(2, 2))
plot(lm_combined)
```



Linear Regression Model Metrics

```
# Function to calculate RMSE
rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

# Calculate RMSE for each model
rmse_category <- rmse(test_data$Average, test_data$Predicted_Avg)
rmse_interaction <- rmse(test_data$Average, test_data$Predicted_Avg_Interaction)
rmse_variety <- rmse(test_data$Average, test_data$Predicted_Avg_Variety)
rmse_combined <- rmse(test_data$Average, test_data$Predicted_Avg_Combined)

# Print RMSE results
cat("RMSE for Category Model:", rmse_category, "\n")
```

RMSE

```
## RMSE for Category Model: 17.64294
```

```
cat("RMSE for Interaction Model:", rmse_interaction, "\n")
```

```
## RMSE for Interaction Model: 17.37907
```

```
cat("RMSE for Variety Model:", rmse_variety, "\n")
```

```
## RMSE for Variety Model: 17.92064
```

```
cat("RMSE for Combined Model:", rmse_combined, "\n")
```

```
## RMSE for Combined Model: 16.76758
```

```r
# Function to calculate MAE
mae <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

# Calculate MAE for each model
mae_category <- mae(test_data$Average, test_data$Predicted_Avg)
mae_interaction <- mae(test_data$Average, test_data$Predicted_Avg_Interaction)
mae_variety <- mae(test_data$Average, test_data$Predicted_Avg_Variety)
mae_combined <- mae(test_data$Average, test_data$Predicted_Avg_Combined)

# Print MAE results
cat("MAE for Category Model:", mae_category, "\n")
```

MAE

```
## MAE for Category Model: 13.66331
```

```r
cat("MAE for Interaction Model:", mae_interaction, "\n")
```

```
## MAE for Interaction Model: 13.57479
```

```r
cat("MAE for Variety Model:", mae_variety, "\n")
```

```
## MAE for Variety Model: 13.94508
```

```r
cat("MAE for Combined Model:", mae_combined, "\n")
```

```
## MAE for Combined Model: 12.81186
```

```r
# Function to calculate R-squared
r_squared <- function(actual, predicted) {
  1 - (sum((actual - predicted)^2) / sum((actual - mean(actual))^2))
}

# Calculate R-squared
r2_category <- r_squared(test_data$Average, test_data$Predicted_Avg)
r2_interaction <- r_squared(test_data$Average, test_data$Predicted_Avg_Interaction)
r2_variety <- r_squared(test_data$Average, test_data$Predicted_Avg_Variety)
r2_combined <- r_squared(test_data$Average, test_data$Predicted_Avg_Combined)

# Print R-squared results
cat("R-squared for Category Model:", r2_category, "\n")
```

R-Squared

```
## R-squared for Category Model: 0.1044897
```

```r
cat("R-squared for Interaction Model:", r2_interaction, "\n")
```

```
## R-squared for Interaction Model: 0.131076
```

```r
cat("R-squared for Variety Model:", r2_variety, "\n")
```

```
## R-squared for Variety Model: 0.07607722
```

```r
cat("R-squared for Combined Model:", r2_combined, "\n")
```

```
## R-squared for Combined Model: 0.1911477
```

```r
# Function to calculate Mean Squared Error (MSE)
mse <- function(actual, predicted) {
  mean((actual - predicted)^2)
}

# Calculate MSE
mse_category <- mse(test_data$Average, test_data$Predicted_Avg)
mse_interaction <- mse(test_data$Average, test_data$Predicted_Avg_Interaction)
mse_variety <- mse(test_data$Average, test_data$Predicted_Avg_Variety)
mse_combined <- mse(test_data$Average, test_data$Predicted_Avg_Combined)


# Print MSE results
cat("MSE Results:\n")
```

MSE

```
## MSE Results:
```

```r
cat("  Category Model:", mse_category, "\n")
```

```
##   Category Model: 311.2733
```

```r
cat("  Interaction Model:", mse_interaction, "\n")
```

```
##   Interaction Model: 302.0321
```

```r
cat("  Variety Model:", mse_variety, "\n")
```

```
##   Variety Model: 321.1493
```

```r
cat("  Combined Model:", mse_combined, "\n\n")
```

```
##   Combined Model: 281.1516
```

```r
# Function to calculate Mean Absolute Percentage Error (MAPE)
mape <- function(actual, predicted) {
  mean(abs((actual - predicted) / actual)) * 100
}

mape_category <- mape(test_data$Average, test_data$Predicted_Avg)
mape_interaction <- mape(test_data$Average, test_data$Predicted_Avg_Interaction)
mape_variety <- mape(test_data$Average, test_data$Predicted_Avg_Variety)
mape_combined <- mape(test_data$Average, test_data$Predicted_Avg_Combined)

# Print MAPE results
cat("MAPE Results:\n")
```

MAPE

```
## MAPE Results:
```

```r
cat("  Category Model:", mape_category, "%\n")
```

```
##   Category Model: 38.46027 %
```

```r
cat("  Interaction Model:", mape_interaction, "%\n")
```

```
##   Interaction Model: 37.62572 %
```

```r
cat("  Variety Model:", mape_variety, "%\n")
```

```
##   Variety Model: 39.70397 %
```

```r
cat("  Combined Model:", mape_combined, "%\n")
```

```
##   Combined Model: 35.03312 %
```

Linear Regression Models Results

```r
# Combine all metrics into a dataframe
results <- data.frame(
  Model = c("Category", "Interaction", "Variety", "Combined"),
  RMSE = c(rmse_category, rmse_interaction, rmse_variety, rmse_combined),
  MAE = c(mae_category, mae_interaction, mae_variety, mae_combined),
  R2 = c(r2_category, r2_interaction, r2_variety, r2_combined),
  MSE = c(mse_category, mse_interaction, mse_variety, mse_combined),
  MAPE = c(mape_category, mape_interaction, mape_variety, mape_combined)
)

# Print the results
print(results)
```

```
##         Model     RMSE      MAE         R2      MSE     MAPE
## 1    Category 17.64294 13.66331 0.10448973 311.2733 38.46027
## 2 Interaction 17.37907 13.57479 0.13107602 302.0321 37.62572
## 3     Variety 17.92064 13.94508 0.07607722 321.1493 39.70397
## 4    Combined 16.76758 12.81186 0.19114765 281.1516 35.03312
```

Stationarity, Decomposition and ACF and PACF

Before fitting my models for ARIMA I performed stationarity test, analyzed the time series decomposition to understand the seasonal patterns and examined the ACF and PACF plots to identify potential AR and MA terms for your ARIMA.

```r
# Filter for different varieties
big_tomato <- data %>% filter(Category == "Tomato Big")
small_tomato <- data %>% filter(Category == "Tomato Small")

# --- Stationarity Testing ---

# Perform ADF test on big_tomato prices
adf_result_big <- adf.test(big_tomato$Average)
```

```
## Warning in adf.test(big_tomato$Average): p-value smaller than printed p-value
```

```r
print(adf_result_big)
```

```
##
##  Augmented Dickey-Fuller Test
```

```
##
## data:  big_tomato$Average
## Dickey-Fuller = -6.0619, Lag order = 14, p-value = 0.01
## alternative hypothesis: stationary
```

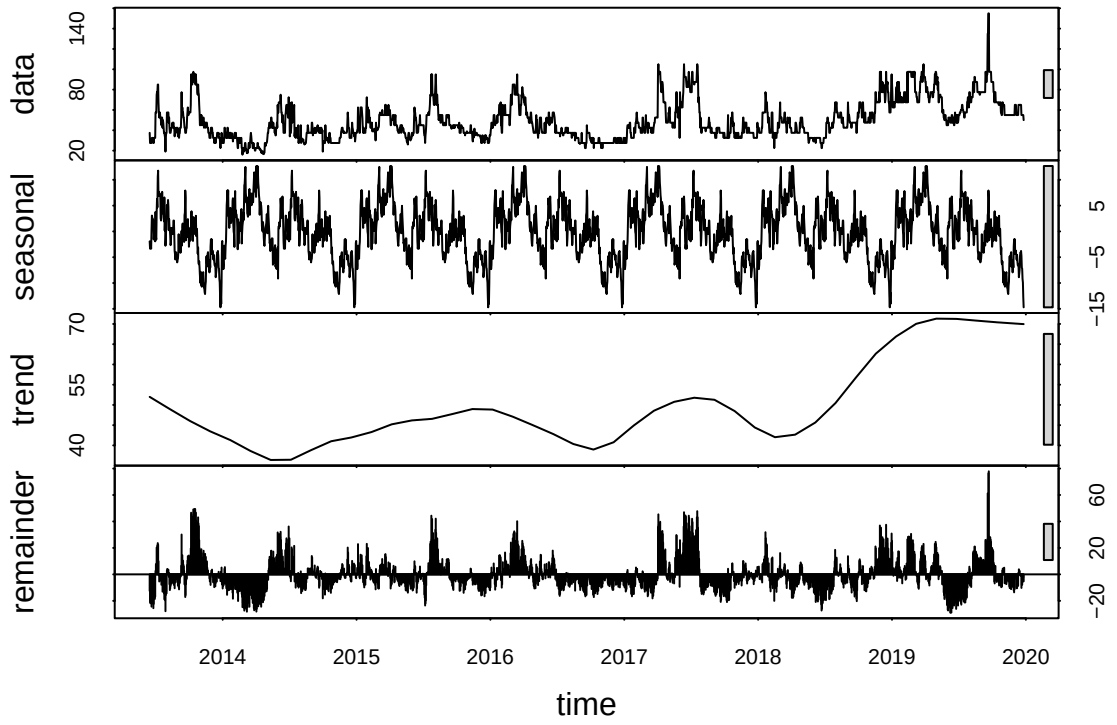
```
# Perform ADF test on small_tomato prices
adf_result_small <- adf.test(small_tomato$Average)
```

```
## Warning in adf.test(small_tomato$Average): p-value smaller than printed p-value
```

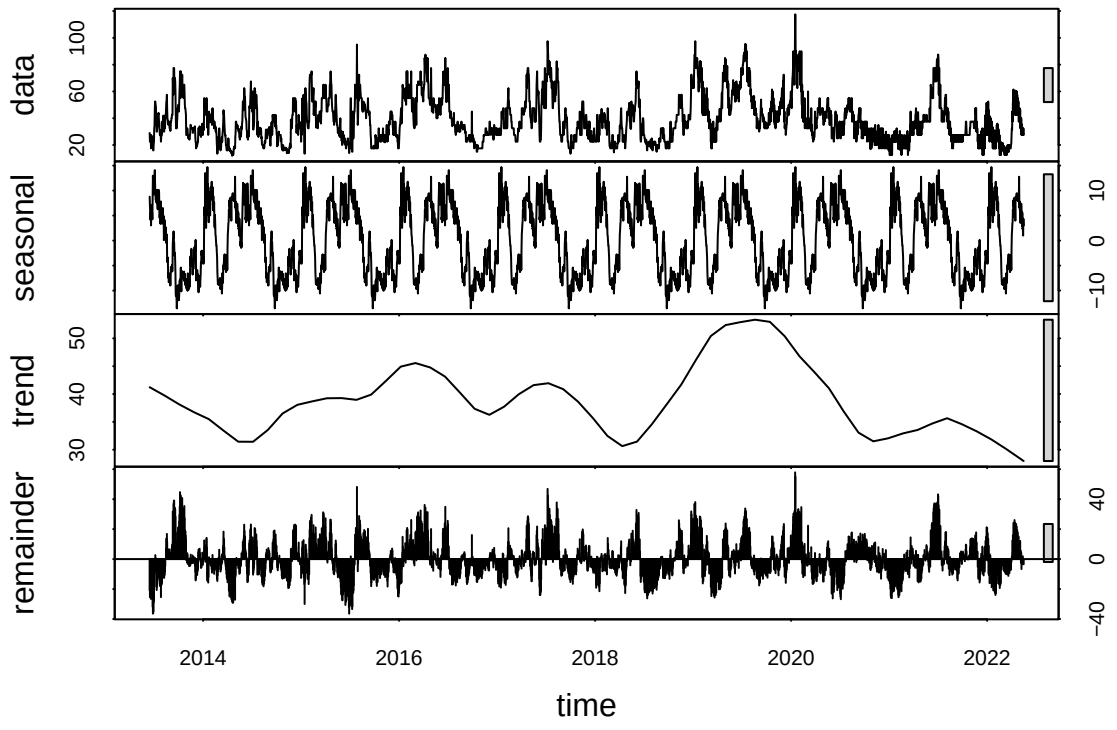
```
print(adf_result_small)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  small_tomato$Average
## Dickey-Fuller = -6.1721, Lag order = 15, p-value = 0.01
## alternative hypothesis: stationary
```

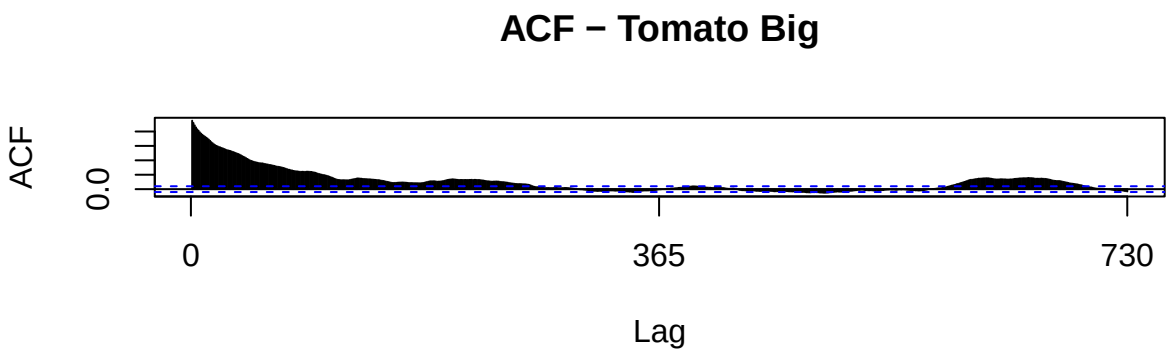
```
# If p-value > 0.05 (non-stationary), apply differencing
# Example for big_tomato (repeat for small_tomato if needed)
if(adf_result_big$p.value > 0.05) {
  big_tomato$Average <- diff(big_tomato$Average)
  # Re-test for stationarity after differencing
  adf_result_big <- adf.test(big_tomato$Average)
  print(adf_result_big)
}


# Determine split index (80% train, 20% test)
split_index_big <- floor(0.8 * nrow(big_tomato))
split_index_small <- floor(0.8 * nrow(small_tomato))

# Split data into train and test sets
train_big <- big_tomato[1:split_index_big, ]
test_big <- big_tomato[(split_index_big + 1):nrow(big_tomato), ]

train_small <- small_tomato[1:split_index_small, ]
test_small <- small_tomato[(split_index_small + 1):nrow(small_tomato), ]

# Create time series objects for training data
ts_train_big <- ts(train_big$Average,
                   start = c(year(min(train_big$Date)), as.numeric(format(min(train_big$Date), "%j"))),
                   frequency = 365)

ts_train_small <- ts(train_small$Average,
                   start = c(year(min(train_small$Date)), as.numeric(format(min(train_small$Date), "%j"
                   frequency = 365)

# Decompose Tomato Big data to observe seasonality
decompose_big <- stl(ts_train_big, s.window = "periodic")
plot(decompose_big)
```

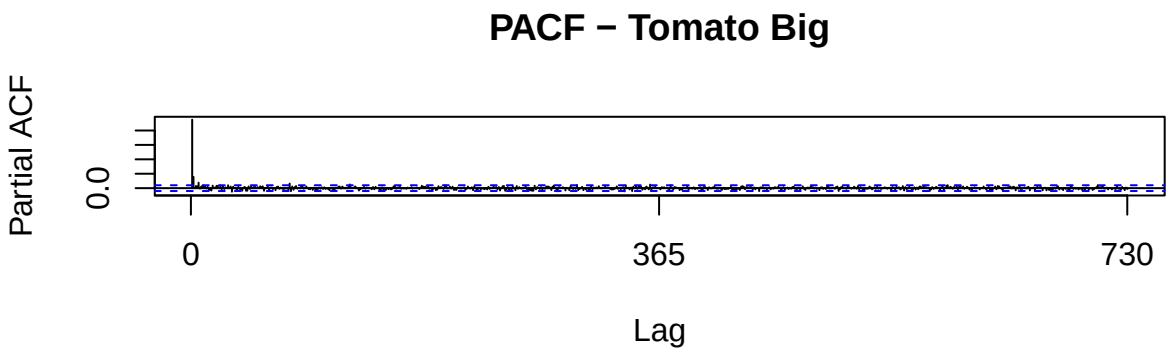



```r
# Decompose Tomato Small data
decompose_small <- stl(ts_train_small, s.window = "periodic")
plot(decompose_small)
```

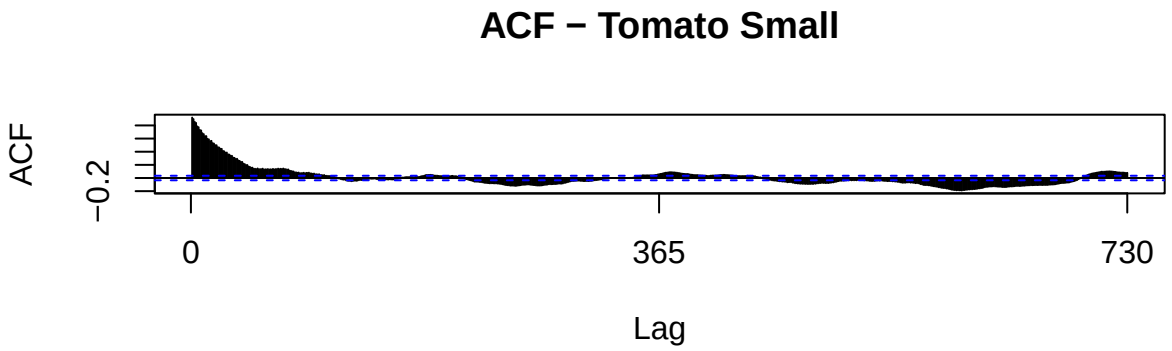


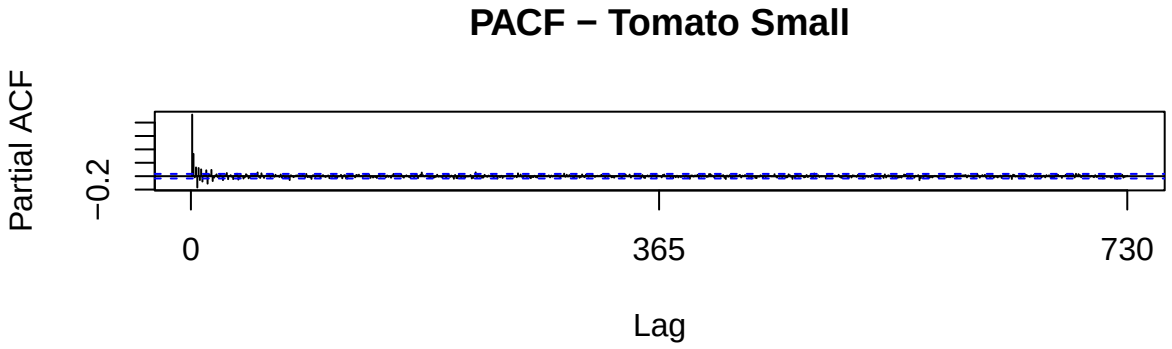

```r
# Check ACF and PACF to identify seasonality
par(mfrow = c(2, 1))
Acf(ts_train_big, main = "ACF - Tomato Big")
Pacf(ts_train_big, main = "PACF - Tomato Big")
```






```
Acf(ts_train_small, main = "ACF - Tomato Small")
Pacf(ts_train_small, main = "PACF - Tomato Small")
```





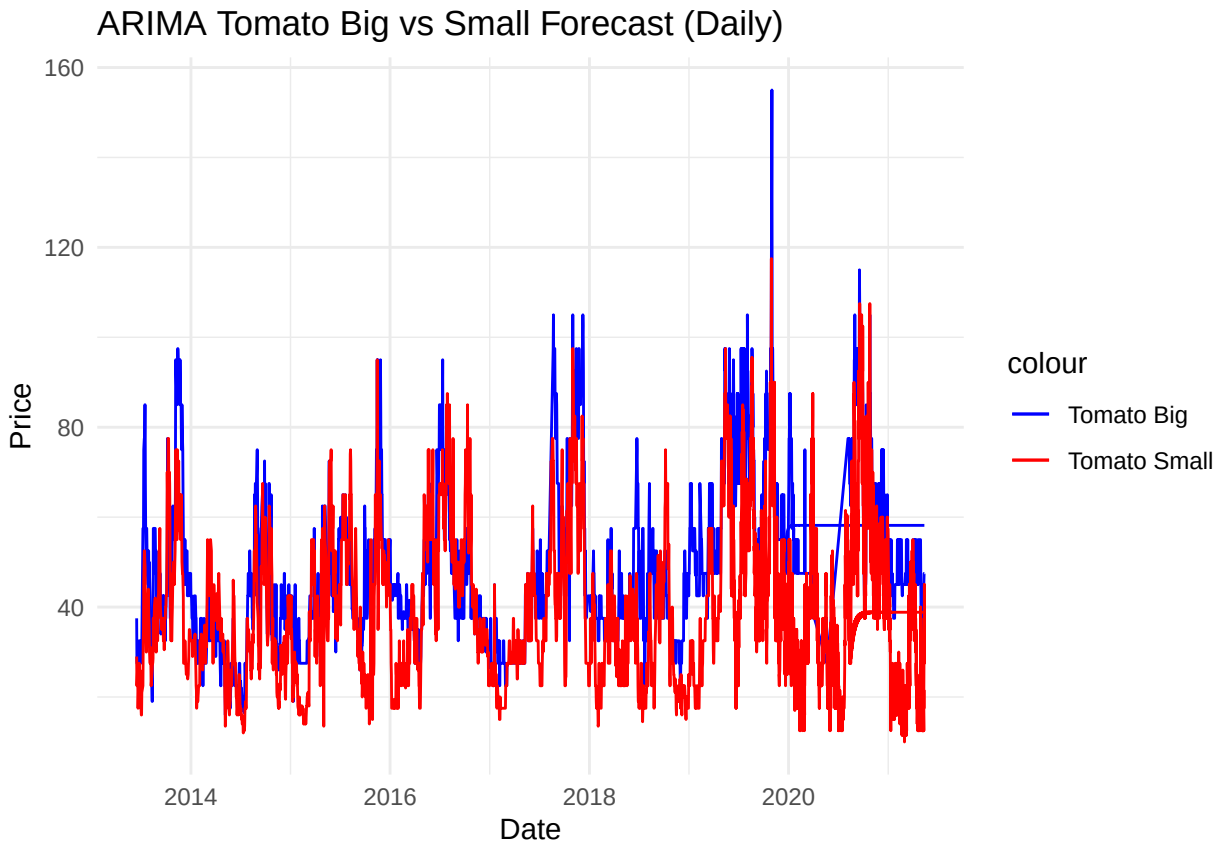
###

ARIMA

```r
# Fit ARIMA models to training data
arima_big <- auto.arima(ts_train_big)
arima_small <- auto.arima(ts_train_small)

# Forecast future prices (on the length of the test set)
forecast_big <- forecast(arima_big, h = nrow(test_big))
forecast_small <- forecast(arima_small, h = nrow(test_small))

# Generate proper date sequence for the entire dataset
big_dates <- as.Date(big_tomato$Date)
small_dates <- as.Date(small_tomato$Date)


# Combine actual values and forecasts with dates
big_forecast_df <- data.frame(
  Date = big_dates,
  Actual = big_tomato$Average,
  Forecast = c(rep(NA, length(ts_train_big)), forecast_big$mean)
)

small_forecast_df <- data.frame(
  Date = small_dates,
  Actual = small_tomato$Average,
  Forecast = c(rep(NA, length(ts_train_small)), forecast_small$mean)
)

# Create and store the ggplot object
arima_tomato_forecast_plot <- ggplot() +
  geom_line(data = big_forecast_df, aes(x = Date, y = Actual, color = "Tomato Big", linetype = "Tomato
  geom_line(data = big_forecast_df, aes(x = Date, y = Forecast, color = "Tomato Big", linetype = "Tomat
  geom_line(data = small_forecast_df, aes(x = Date, y = Actual, color = "Tomato Small", linetype = "Toma
  geom_line(data = small_forecast_df, aes(x = Date, y = Forecast, color = "Tomato Small", linetype = "T
  labs(title = "ARIMA Tomato Big vs Small Forecast (Daily)", x = "Date", y = "Price") +
  theme_minimal() +
  scale_color_manual(
    values = c("Tomato Big" = "blue", "Tomato Small" = "red")
  ) +
  scale_linetype_manual(
    values = c("Tomato Big" = "solid", "Tomato Big" = "dashed",
               "Tomato Small" = "solid", "Tomato Small" = "dashed")
  ) +
  guides(linetype = "none")  # Remove the linetype legend

print(arima_tomato_forecast_plot)
```

```
## Warning: Removed 2384 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 3256 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



ARIMA Metrics

```
# Evaluate ARIMA for Big Tomato
actual_big <- test_big$Average
predicted_big <- forecast_big$mean

rmse_arima_big <- rmse(actual_big, predicted_big)
mae_arima_big <- mae(actual_big, predicted_big)
mse_arima_big <- mse(actual_big, predicted_big)
mape_arima_big <- mape(actual_big, predicted_big)

cat("Tomato Big - RMSE:", rmse_arima_big, "\n")
```

```
## Tomato Big - RMSE: 14.0819
```

```
cat("Tomato Big - MAE:", mae_arima_big, "\n")
```

```
## Tomato Big - MAE: 11.40682
```

```
cat("Tomato Big - MSE:", mse_arima_big, "\n")
```

```
## Tomato Big - MSE: 198.2999
```

```
cat("Tomato Big - MAPE:", mape_arima_big, "\n")
```

```
## Tomato Big - MAPE: 20.59743
```

```
# Evaluate ARIMA for Small Tomato
actual_small <- test_small$Average
predicted_small <- forecast_small$mean
```

```r
rmse_arima_small <- rmse(actual_big, predicted_big)
mae_arima_small <- mae(actual_big, predicted_big)
mse_arima_small <- mse(actual_big, predicted_big)
mape_arima_small <- mape(actual_big, predicted_big)

cat("Tomato Small - RMSE:", rmse_arima_small, "\n")
```

```
## Tomato Small - RMSE: 14.0819
```

```r
cat("Tomato Small - MAE:", mape_arima_small, "\n")
```

```
## Tomato Small - MAE: 20.59743
```

```r
cat("Tomato Small - MSE:", mse_arima_small, "\n")
```

```
## Tomato Small - MSE: 198.2999
```

```r
cat("Tomato Small - MAPE:", mape_arima_small, "\n")
```

```
## Tomato Small - MAPE: 20.59743
```

SARIMA

```r
# Fit SARIMA model for Tomato Big
sarima_big <- auto.arima(ts_train_big, seasonal = TRUE, stepwise = FALSE, approximation = FALSE)

# Summary of the model
summary(sarima_big)
```

```
## Series: ts_train_big
## ARIMA(2,1,1)
##
## Coefficients:
##          ar1     ar2      ma1
##       0.7834  0.1444  -0.9872
## s.e.  0.0613  0.0477   0.0540
##
## sigma^2 = 32.57:  log likelihood = -7530.73
## AIC=15069.46   AICc=15069.47   BIC=15092.56
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.05360092 5.702517 3.414604 -1.004343 7.044579 0.1792371
##                      ACF1
## Training set 0.0005907794
```

```r
# Fit SARIMA model for Tomato Small
sarima_small <- auto.arima(ts_train_small, seasonal = TRUE, stepwise = FALSE, approximation = FALSE)

# Summary of the model
summary(sarima_small)
```

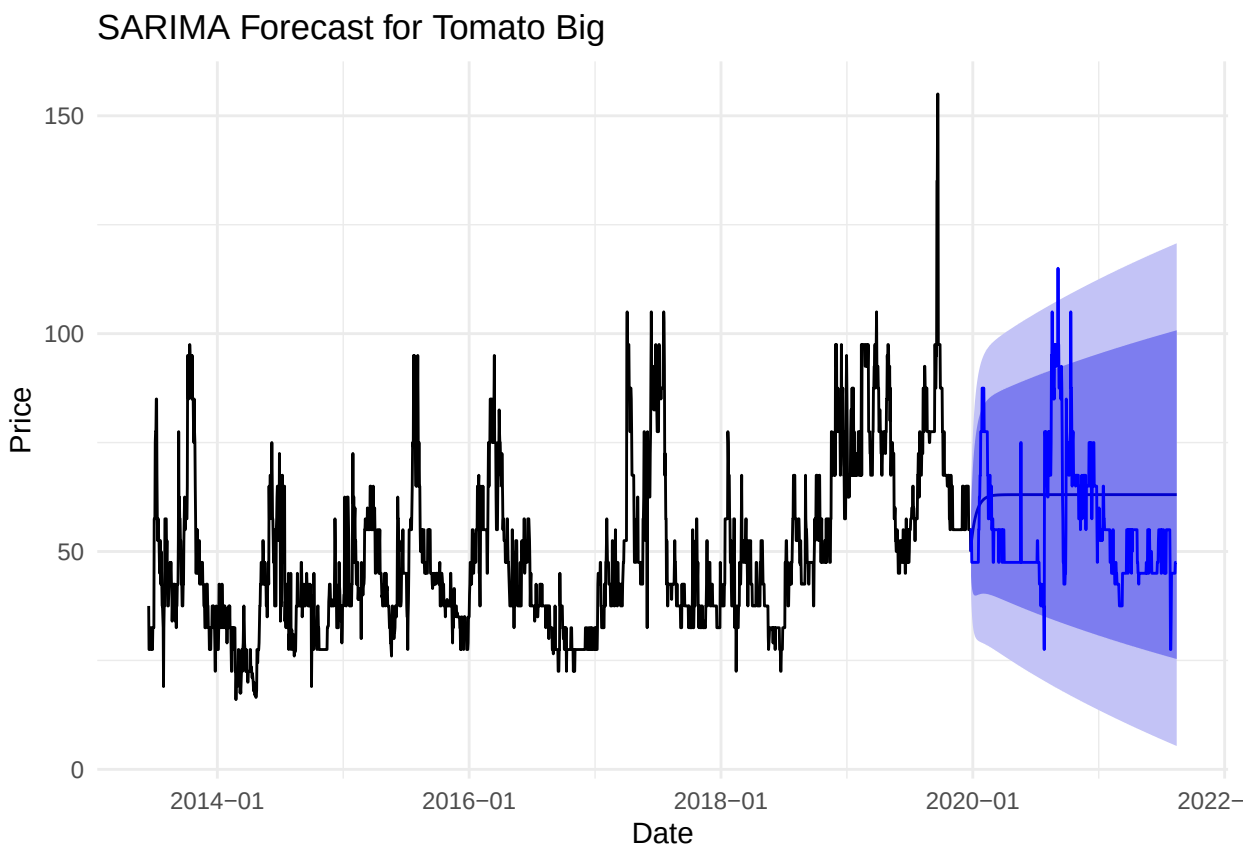
```
## Series: ts_train_small
## ARIMA(4,0,1) with non-zero mean
##
## Coefficients:
##          ar1     ar2     ar3     ar4     ma1     mean
```

```
##       -0.1942  0.7946  0.1692  0.1389  0.8893  38.8481
## s.e.   0.0223  0.0206  0.0196  0.0188  0.0152   2.1209
##
## sigma^2 = 34.97:  log likelihood = -10404.92
## AIC=20823.85   AICc=20823.88   BIC=20866.47
##
## Training set error measures:
##                         ME    RMSE      MAE       MPE     MAPE      MASE
## Training set -0.0005504609 5.90771 4.145752 -2.440516 11.49476 0.2380742
##                      ACF1
## Training set 0.004202936
```

```
forecast_big_sarima <- forecast(sarima_big, h = nrow(test_big))
# Convert test set average prices to a time series
test_big_ts <- ts(test_big$Average, start = end(ts_train_big), frequency = 365)

# Plot forecast
autoplot(forecast_big_sarima) +
  autolayer(test_big_ts, series = "Actual", color = "blue") +
  ggtitle("SARIMA Forecast for Tomato Big") +
  xlab("Date") + ylab("Price") +
  theme_minimal() +
  scale_color_manual(values = c("Actual" = "blue", "Forecast" = "black")) +
  scale_x_yearmon(format = "%Y-%m")
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```

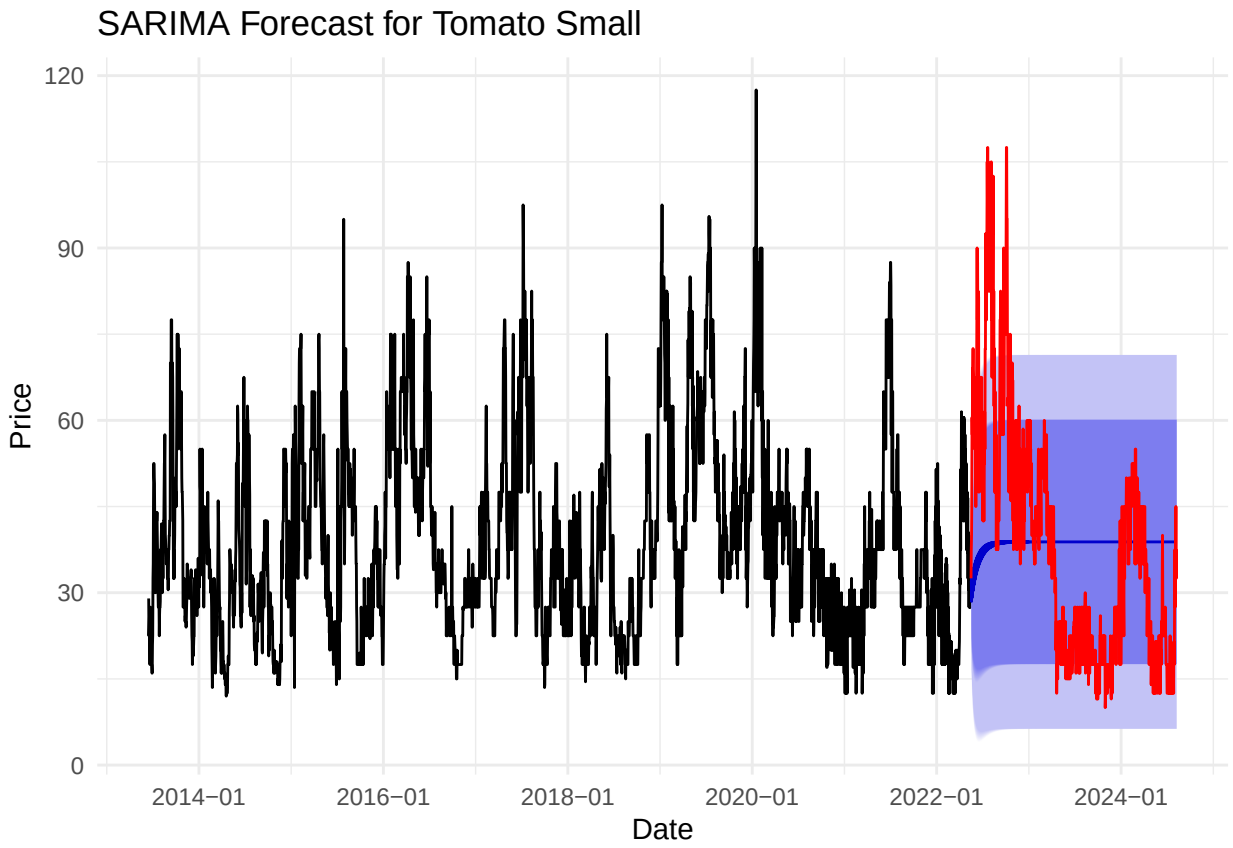
```
## Warning: No shared levels found between `names(values)` of the manual scale and the
## data's colour values.
```




```
forecast_small_sarima <- forecast(sarima_small, h = nrow(test_small))
# Convert test set average prices to a time series
test_small_ts <- ts(test_small$Average, start = end(ts_train_small), frequency = 365)


# Plot forecast
autoplot(forecast_small_sarima) +
  autolayer(test_small_ts, series = "Actual", color = "red") +
  ggtitle("SARIMA Forecast for Tomato Small") +
  xlab("Date") + ylab("Price") +
  theme_minimal() +
  scale_color_manual(values = c("Actual" = "red", "Forecast" = "black"))+
  scale_x_yearmon(format = "%Y-%m")
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```

```
## Warning: No shared levels found between `names(values)` of the manual scale and the
## data's colour values.
```



SARIMA Metrics

```
# Tomato Big
actual_big <- test_big$Average
predicted_big <- forecast_big_sarima$mean

rmse_sarima_big <- rmse(actual_big, predicted_big)
mae_sarima_big <- mae(actual_big, predicted_big)
mse_sarima_big <- mse(actual_big, predicted_big)
mape_sarima_big <- mape(actual_big, predicted_big)

cat("Tomato Big - RMSE:", rmse_sarima_big, "\n")
```

```
## Tomato Big - RMSE: 15.42693
```

```
cat("Tomato Big - MAE:", mae_sarima_big, "\n")
```

```
## Tomato Big - MAE: 13.45333
```

```
cat("Tomato Big - MSE:", mse_sarima_big, "\n")
```

```
## Tomato Big - MSE: 237.9903
```

```
cat("Tomato Big - MAPE:", mape_sarima_big, "\n")
```

```
## Tomato Big - MAPE: 25.79545
```

```
# Tomato Small
actual_small <- test_small$Average
```

```r
predicted_small <- forecast_small_sarima$mean

rmse_sarima_small <- rmse(actual_big, predicted_big)
mae_sarima_small <- mae(actual_big, predicted_big)
mse_sarima_small <- mse(actual_big, predicted_big)
mape_sarima_small <- mape(actual_big, predicted_big)

cat("Tomato Small - RMSE:", rmse_sarima_small, "\n")
```

```
## Tomato Small - RMSE: 15.42693
```

```r
cat("Tomato Small - MAE:", mape_sarima_small, "\n")
```

```
## Tomato Small - MAE: 25.79545
```

```r
cat("Tomato Small - MSE:", mse_sarima_small, "\n")
```

```
## Tomato Small - MSE: 237.9903
```

```r
cat("Tomato Small - MAPE:", mape_sarima_small, "\n")
```

```
## Tomato Small - MAPE: 25.79545
```

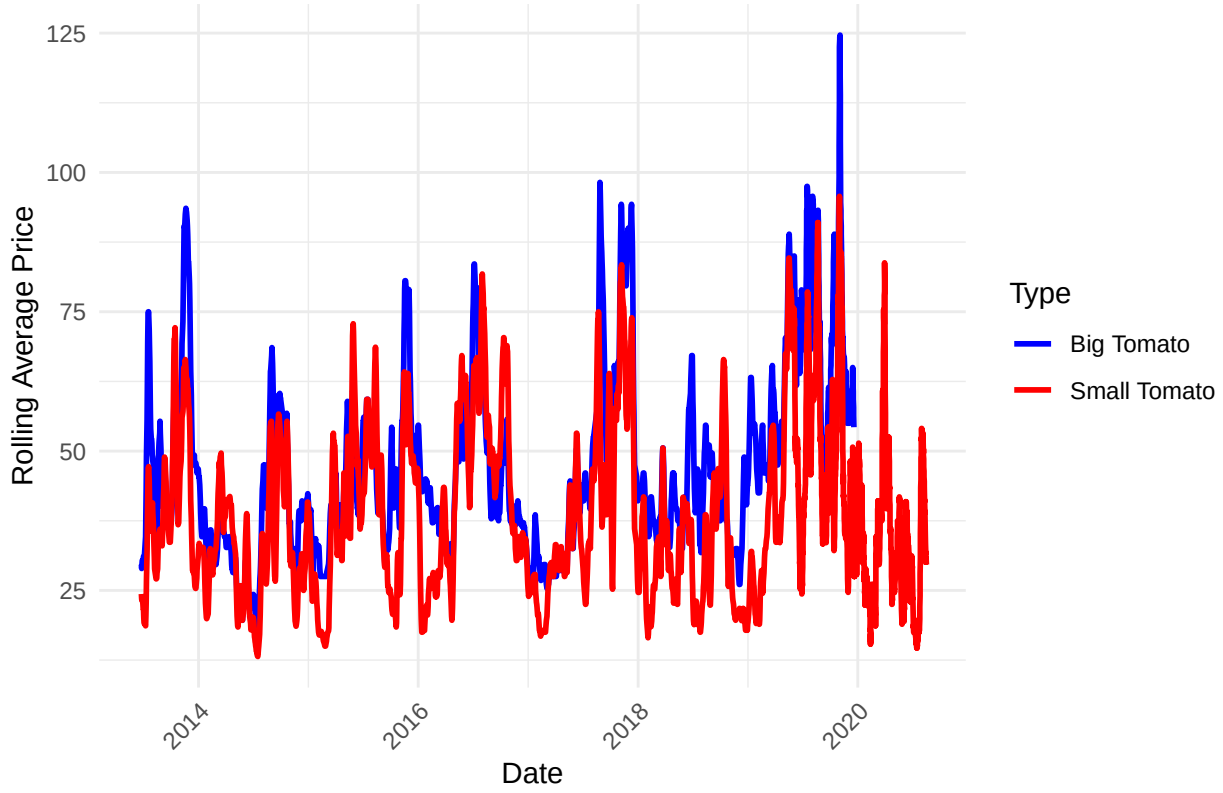
Rolling Average

```r
# Apply rolling mean with a 7-day window to training data
train_big$Rolling_Avg <- rollmean(train_big$Average, k = 7, fill = NA, align = "right")
train_small$Rolling_Avg <- rollmean(train_small$Average, k = 7, fill = NA, align = "right")

# Combine rolling averages for plotting
rolling_data <- rbind(
  data.frame(Date = train_big$Date, Rolling_Avg = train_big$Rolling_Avg, Type = "Big Tomato"),
  data.frame(Date = train_small$Date, Rolling_Avg = train_small$Rolling_Avg, Type = "Small Tomato")
)

# Plot rolling averages
rolling_averages_plot <- ggplot(rolling_data, aes(x = Date, y = Rolling_Avg, color = Type)) +
  geom_line(size = 1) +
  labs(
    title = "7-Day Rolling Average: Big vs Small Tomato (Training Data)",  # Updated title
    x = "Date",
    y = "Rolling Average Price"
  ) +
  theme_minimal() +
  scale_color_manual(values = c("Big Tomato" = "blue", "Small Tomato" = "red")) +
  theme(
    plot.title = element_text(size = 14, face = "bold", hjust = 0.5),
    axis.text.x = element_text(angle = 45, hjust = 1)  # Rotate x-axis for better readability
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
print(rolling_averages_plot)
```

```
## Warning: Removed 12 rows containing missing values or values outside the scale range
## (`geom_line()`).
```




```
# Save the plot to a file
```

VAR and VECM Model and Cointegration

```
# Align and Clean Data
aligned_data <- merge(
  big_tomato[, c("Date", "Average")],
  small_tomato[, c("Date", "Average")],
  by = "Date", suffixes = c("_Big", "_Small")
)
aligned_data$Average_Big <- na.approx(aligned_data$Average_Big)
aligned_data$Average_Small <- na.approx(aligned_data$Average_Small)

# Prepare Data Matrix
data_matrix <- as.matrix(aligned_data[, c("Average_Big", "Average_Small")])

# Johansen Cointegration Test
johansen_test <- ca.jo(data_matrix, type = "trace", ecdet = "const", K = 2)
summary(johansen_test)
```

```
##
## ######################
```

```
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , without linear trend and constant in cointegration
##
## Eigenvalues (lambda):
## [1] 3.101881e-02 1.219555e-02 1.040834e-17
##
## Values of teststatistic and critical values of test:
##
##            test 10pct  5pct  1pct
## r <= 1 |  61.06  7.52  9.24 12.97
## r = 0  | 217.85 17.85 19.96 24.60
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##                  Average_Big.l2 Average_Small.l2     constant
## Average_Big.l2          1.00000        1.0000000    1.0000000
## Average_Small.l2       -1.05355        0.6649829    0.2994264
## constant              -12.14594      -79.6217729 2248.9944347
##
## Weights W:
## (This is the loading matrix)
##
##                 Average_Big.l2 Average_Small.l2      constant
## Average_Big.d      -0.02742881      -0.01607258  6.081350e-20
## Average_Small.d     0.05336442      -0.01787248 -3.067917e-19
```

```r
# Cointegration Check
test_stat <- johansen_test@teststat
critical_val <- johansen_test@cval[, "5pct"]

if (test_stat[1] > critical_val[1]) {
  cat("\nCointegration detected: Proceeding with VECM...\n")

  # VECM Model and Forecasting
  vecm_as_var <- vec2var(johansen_test, r = 1)
  n_ahead <- 10
  vecm_forecast <- predict(vecm_as_var, n.ahead = n_ahead)

  # Back-Transform Forecasts
  last_big_value <- tail(aligned_data$Average_Big, 1)
  last_small_value <- tail(aligned_data$Average_Small, 1)

  forecast_big <- cumsum(c(last_big_value, vecm_forecast$fcst$Average_Big[, "fcst"]))[-1]
  forecast_small <- cumsum(c(last_small_value, vecm_forecast$fcst$Average_Small[, "fcst"]))[-1]

  # Forecast Data Frame
  forecast_dates <- seq(as.Date(tail(aligned_data$Date, 1)) + 1, by = "days", length.out = n_ahead)
  forecast_data <- data.frame(
    Date = forecast_dates,
    Big_Tomato_Forecast = forecast_big,
    Small_Tomato_Forecast = forecast_small
```

```
  )

  # Metrics (Using last n_ahead actuals as dummy for comparison)
  actual_big <- tail(aligned_data$Average_Big, n_ahead)
  actual_small <- tail(aligned_data$Average_Small, n_ahead)

  rmse_vecm_big <- rmse(actual_big, predicted_big)
  mae_vecm_big <- mae(actual_big, predicted_big)
  mse_vecm_big <- mse(actual_big, predicted_big)
  mape_vecm_big <- mape(actual_big, predicted_big)


  rmse_vecm_small <- rmse(actual_big, predicted_big)
  mae_vecm_small <- mae(actual_big, predicted_big)
  mse_vecm_small <- mse(actual_big, predicted_big)
  mape_vecm_small <- mape(actual_big, predicted_big)


  cat("\n--- Metrics for VECM Forecast ---\n")
  cat("Big Tomato RMSE:", rmse_vecm_big, "\n")
  cat("Big Tomato MAE:", mae_vecm_big, "\n")
  cat("Big Tomato RMSE:", mse_vecm_big, "\n")
  cat("Big Tomato MAE:", mape_vecm_big, "\n")
  cat("Small Tomato RMSE:", rmse_vecm_small, "\n")
  cat("Small Tomato MAE:", mae_vecm_small, "\n")
  cat("Small Tomato MSE:", mse_vecm_small, "\n")
  cat("Small Tomato MAPE:", mape_vecm_small, "\n")


  # Plot Forecast Results for VECM
  vecm_plot <- ggplot(forecast_data, aes(x = Date)) +
    geom_line(aes(y = Big_Tomato_Forecast, color = "Big Tomato (Forecast)"), size = 1) +
    geom_line(aes(y = Small_Tomato_Forecast, color = "Small Tomato (Forecast)"), size = 1) +
    labs(
      title = "VECM Forecast: Big vs Small Tomatoes",
      x = "Date",
      y = "Prices"
    ) +
    theme_minimal() +
    scale_color_manual(
      name = "Legend",
      values = c("Big Tomato (Forecast)" = "blue", "Small Tomato (Forecast)" = "red")
    )
  print(vecm_plot)

} else {
  cat("\nNo Cointegration detected: Proceeding with VAR model...\n")

  # Train-Test Split
  train_ratio <- 0.8
  split_index <- floor(nrow(aligned_data) * train_ratio)
  train_data <- aligned_data[1:split_index, c("Average_Big", "Average_Small")]
  test_data <- aligned_data[(split_index + 1):nrow(aligned_data), ]
```

```r
# Fit VAR Model
lag_selection <- VARselect(train_data, lag.max = 10, type = "const")
var_model <- VAR(train_data, p = lag_selection$selection["AIC(n)"], type = "const")
n_ahead <- nrow(test_data)

# Forecast using VAR Model
var_forecast <- predict(var_model, n.ahead = n_ahead)

# Back-Transform Forecasts
last_big_value <- tail(aligned_data$Average_Big[1:split_index], 1)
last_small_value <- tail(aligned_data$Average_Small[1:split_index], 1)

forecast_big <- cumsum(c(last_big_value, var_forecast$fcst$Average_Big[, "fcst"]))[-1]
forecast_small <- cumsum(c(last_small_value, var_forecast$fcst$Average_Small[, "fcst"]))[-1]

# Forecast Data Frame
forecast_dates <- test_data$Date
forecast_data <- data.frame(
  Date = forecast_dates,
  Big_Tomato_Forecast = forecast_big,
  Small_Tomato_Forecast = forecast_small
)

# Metrics
actual_big <- test_data$Average_Big
actual_small <- test_data$Average_Small

rmse_vecm_big <- rmse(actual_big, predicted_big)
mae_vecm_big <- mae(actual_big, predicted_big)
mse_vecm_big <- mse(actual_big, predicted_big)
mape_vecm_big <- mape(actual_big, predicted_big)


rmse_vecm_small <- rmse(actual_big, predicted_big)
mae_vecm_small <- mae(actual_big, predicted_big)
mse_vecm_small <- mse(actual_big, predicted_big)
mape_vecm_small <- mape(actual_big, predicted_big)


cat("\n--- Metrics for VECM Forecast ---\n")
cat("Big Tomato RMSE:", rmse_vecm_big, "\n")
cat("Big Tomato MAE:", mae_vecm_big, "\n")
cat("Big Tomato RMSE:", mse_vecm_big, "\n")
cat("Big Tomato MAE:", mape_vecm_big, "\n")
cat("Small Tomato RMSE:", rmse_vecm_small, "\n")
cat("Small Tomato MAE:", mae_vecm_small, "\n")
cat("Small Tomato MSE:", mse_vecm_small, "\n")
cat("Small Tomato MAPE:", mape_vecm_small, "\n")


# Plot Forecast Results for VAR
var_model <- ggplot(forecast_data, aes(x = Date)) +
  geom_line(aes(y = Big_Tomato_Forecast, color = "Big Tomato (Forecast)"), size = 1) +
```

```
    geom_line(aes(y = Small_Tomato_Forecast, color = "Small Tomato (Forecast)"), size = 1) +
    labs(
      title = "VAR Forecast: Big vs Small Tomatoes",
      x = "Date",
      y = "Prices"
    ) +
    theme_minimal() +
    scale_color_manual(
      name = "Legend",
      values = c("Big Tomato (Forecast)" = "blue", "Small Tomato (Forecast)" = "red")
    )

  print(varm_plot)
}
```

```
##
## Cointegration detected: Proceeding with VECM...
```

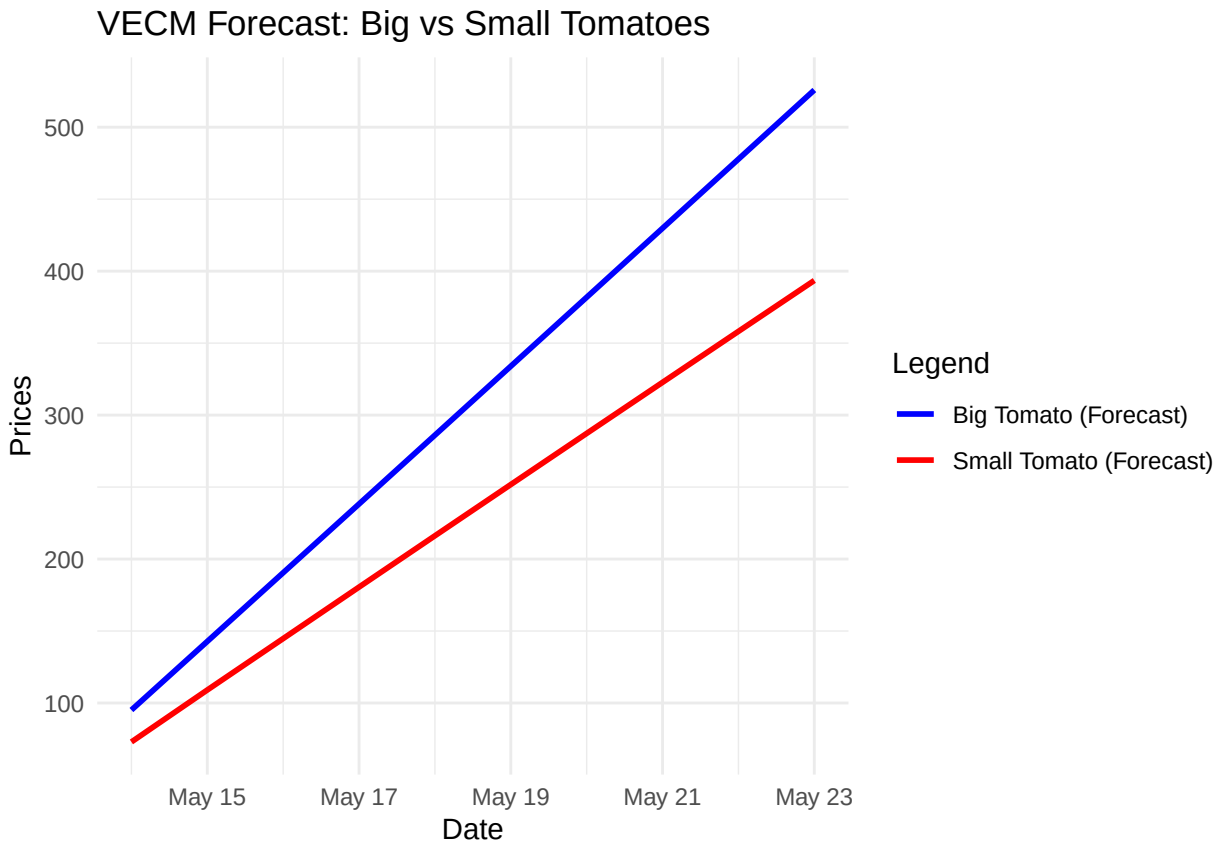
```
## Warning in `-.default`(actual, predicted): longer object length is not a
## multiple of shorter object length
## Warning in `-.default`(actual, predicted): longer object length is not a
## multiple of shorter object length
## Warning in `-.default`(actual, predicted): longer object length is not a
## multiple of shorter object length
## Warning in `-.default`(actual, predicted): longer object length is not a
## multiple of shorter object length
```

```
## Warning in `/.default`((actual - predicted), actual): longer object length is
## not a multiple of shorter object length
```

```
## Warning in `-.default`(actual, predicted): longer object length is not a
## multiple of shorter object length
## Warning in `-.default`(actual, predicted): longer object length is not a
## multiple of shorter object length
## Warning in `-.default`(actual, predicted): longer object length is not a
## multiple of shorter object length
## Warning in `-.default`(actual, predicted): longer object length is not a
## multiple of shorter object length
```

```
## Warning in `/.default`((actual - predicted), actual): longer object length is
## not a multiple of shorter object length
```

```
##
## --- Metrics for VECM Forecast ---
## Big Tomato RMSE: 15.29584
## Big Tomato MAE: 15.23921
## Big Tomato RMSE: 233.9627
## Big Tomato MAE: 32.08255
## Small Tomato RMSE: 15.29584
## Small Tomato MAE: 15.23921
## Small Tomato MSE: 233.9627
## Small Tomato MAPE: 32.08255
```




```r
# Function: Ensure Date Format
prepare_data <- function(data, column_name) {
  data$Date <- as.Date(data$Date)
  return(data)
}

# Function: Stationarity Check and Differencing
stationarity_check <- function(data, column) {
  adf_result <- adf.test(data[[column]], k = 0)
  if (adf_result$p.value > 0.05) {
    cat("Differencing data...\n")
    differenced_data <- diff(data[[column]], differences = 1)
    differenced_data <- na.omit(differenced_data)
  } else {
    differenced_data <- data[[column]]
  }
  return(differenced_data)
}

# Function: Train-Test Split
train_test_split <- function(data, test_size) {
  list(train = head(data, -test_size), test = tail(data, test_size))
}

# Function: Fit GARCH Model
```

```r
fit_garch <- function(train_data, test_size) {
  spec <- ugarchspec(
    variance.model = list(model = "sGARCH", garchOrder = c(1, 1)),
    mean.model = list(armaOrder = c(1, 0), include.mean = TRUE),
    distribution.model = "norm"
  )
  fit <- ugarchfit(spec = spec, data = train_data, out.sample = test_size)
  return(fit)
}

# Function: Forecast GARCH Model
forecast_garch <- function(fit, test_size) {
  forecast <- ugarchforecast(fit, n.ahead = test_size, n.roll = test_size - 1)
  list(mean = as.numeric(fitted(forecast)), volatility = as.numeric(sigma(forecast)))
}

# Function: Evaluate Forecasts
evaluate_forecasts <- function(actual, forecast) {
  cat("RMSE:", rmse(actual, forecast), "\n")
  cat("MAE:", mae(actual, forecast), "\n")
  cat("MSE:", mse(actual, forecast), "\n")
  cat("MAPE:", mape(actual, forecast), "\n")
}

# Function: Plot Results
plot_forecast <- function(dates, actual, forecast_mean, forecast_volatility, title) {
  forecast_data <- data.frame(Date = dates, Actual = actual, Forecast_Mean = forecast_mean, Forecast_Vol

  garch_plot <- ggplot(forecast_data, aes(x = Date)) +
    geom_line(aes(y = Actual, color = "Actual Prices"), size = 1) +
    geom_line(aes(y = Forecast_Mean, color = "Forecasted Mean"), linetype = "dashed", size = 1) +
    geom_line(aes(y = Forecast_Volatility, color = "Forecasted Volatility"), linetype = "dotted", size =
    labs(title = title, x = "Date", y = "Price / Volatility") +
    scale_color_manual(values = c("Actual Prices" = "blue", "Forecasted Mean" = "red", "Forecasted Volat
    theme_minimal()

  print(garch_plot)
}

# Main Workflow for GARCH Analysis
garch_workflow <- function(data, column, test_size, title) {
  data <- prepare_data(data, column)
  differenced_data <- stationarity_check(data, column)
  split <- train_test_split(differenced_data, test_size)

  garch_fit <- fit_garch(split$train, test_size)
  forecast <- forecast_garch(garch_fit, test_size)

  actual_values <- tail(data[[column]], test_size)
  cat("\n--- Metrics ---\n")
  evaluate_forecasts(actual_values, forecast$mean)

  plot_forecast(tail(data$Date, test_size), actual_values, forecast$mean, forecast$volatility, title)
```

```
}

# --- Run for Big and Small Tomatoes ---
cat("\n--- Big Tomato Analysis ---\n")
```

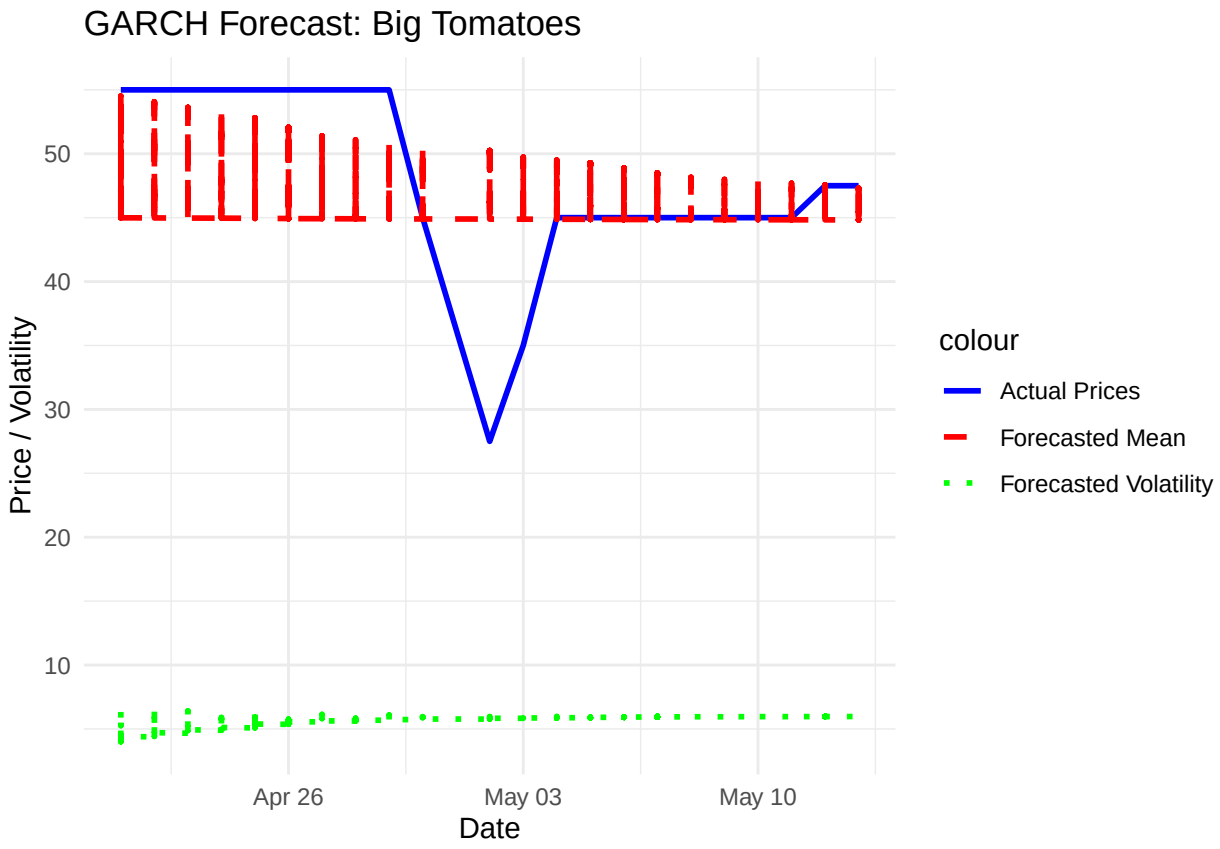
GARCH Models

```
##
## --- Big Tomato Analysis ---
```

```
garch_workflow(big_tomato, "Average", test_size = 30, title = "GARCH Forecast: Big Tomatoes")
```

```
## Warning in adf.test(data[[column]], k = 0): p-value smaller than printed
## p-value
```

```
##
## --- Metrics ---
## RMSE: 7.348163
## MAE: 4.953835
## MSE: 53.9955
## MAPE: 12.24883
```

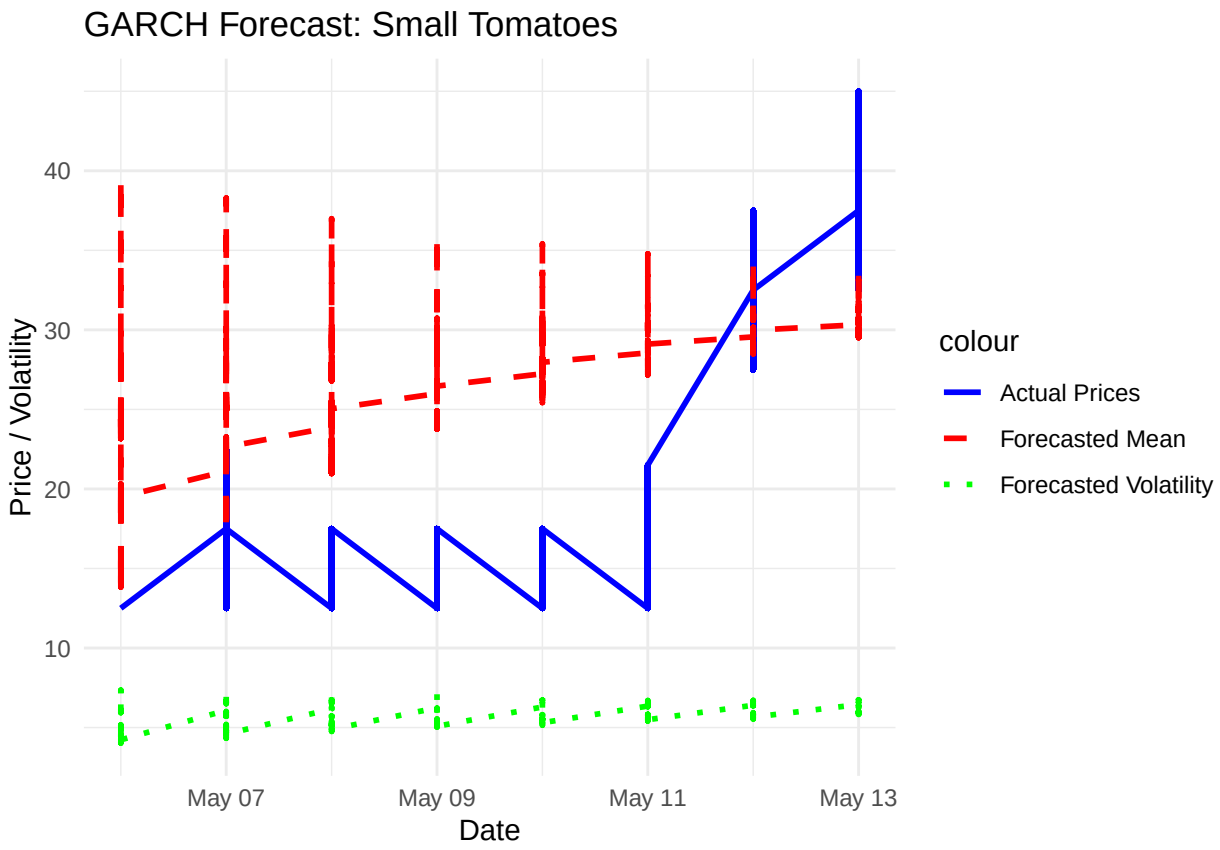



```
cat("\n--- Small Tomato Analysis ---\n")
```

```
##
## --- Small Tomato Analysis ---
```

```
small_tomato <- data %>% filter(grepl("Small", Commodity))
garch_workflow(small_tomato, "Average", test_size = 30, title = "GARCH Forecast: Small Tomatoes")
```

```
## Warning in adf.test(data[[column]], k = 0): p-value smaller than printed
## p-value

##
## --- Metrics ---
## RMSE: 10.75728
## MAE: 9.487709
## MSE: 115.7192
## MAPE: 58.19615
```



Benchmarking

```
results <- data.frame(
  Model = c("Linear Regression Category", "Linear Regression Interaction", "Linear Regression Variety",
  RMSE = c(rmse_category, rmse_interaction, rmse_variety, rmse_combined, rmse_arima_big, rmse_arima_smal
  MAE = c(mae_category, mae_interaction, mae_variety, mae_combined, mae_arima_big, mae_arima_small, mae
  MSE = c(mse_category, mse_interaction, mse_variety, mse_combined, mse_arima_big, mse_arima_small, mse
  MAPE = c(mape_category, mape_interaction, mape_variety, mape_combined, mape_arima_big, mape_arima_smal
)

print(results)
```

```
##                           Model      RMSE       MAE      MSE     MAPE
## 1     Linear Regression Category 17.642938 13.663305 311.2733 38.46027
## 2  Linear Regression Interaction 17.379070 13.574792 302.0321 37.62572
## 3      Linear Regression Variety 17.920638 13.945077 321.1493 39.70397
## 4     Linear Regression Combined 16.767575 12.811861 281.1516 35.03312
## 5                ARIMA Tomato Big 14.081899 11.406822 198.2999 20.59743
## 6              ARIMA Tomato Small 14.081899 11.406822 198.2999 20.59743
```

```
## 7              SARIMA Tomato Big 15.426933 13.453333 237.9903 25.79545
## 8              SARIMA Tomato Big 15.426933 13.453333 237.9903 25.79545
## 9                VECM Tomato Big 15.295840 15.239209 233.9627 32.08255
## 10             VECM Tomato Small 15.295840 15.239209 233.9627 32.08255
## 11              GARCH Tomato Big  7.348163  4.953835  53.9955 12.24883
## 12           GARCH Tomato Small 10.757280  9.487709 115.7192 58.19615
```